

- 3-D Graphics
- A Peek at Taligent

November/December 1994
Display Until 12/31/94

THIS
ISSUE:

**GUI
Tools**

os/2 Developer

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

**COLOR PALETTE
MANAGEMENT**

**OBJECT PERSISTENCE
WITH C++**

**Task Scheduling
on OS/2**

CUSTOMIZING LINES

OpenGL on OS/2

**BUYER'S GUIDE:
GUI Tools**

**THE
ART OF
GRAPHICS
PROGRAMMING**

U.S. \$9.95 Vol. 6 No. 6



0 74476

A Miller Freeman Publication

Warp Speed



SMART redefines migration technology. Version 1.0 made porting feasible. The new version 2.0 makes it IRRESISTIBLE. It's the only tool available to launch your application to OS/2 at Warp Speed. But that's what you've come to expect from One Up Corporation, recipient of the 1994 OS/2 Professional "CORPORATE COMMITMENT AWARD".

What's Next?

Redefinition of OS/2 education, marketing and conversion assistance . . .

Expect The Unexpected!

One UP
Corporation

1603 LBJ Freeway, Suite 200 • Dallas, Texas 75234

1-800-678-01UP

FAX (214) 620-9626

Circle Reader Service Number 1

WHICH PATH TO SOM WOULD YOU RATHER TAKE?

[*The typical journey with IDL*]



OR



[*MetaWare's DirectToSOM alternative*]

Find your way to binary-compatible C++ objects, and skip the IDL detour: Go DirectToSOM with MetaWare's High C/C++™. High C/C++ allows you to inherit Workplace Shell SOM classes just as you would any other C++ class—without IDL. You can use High C/C++ from within Workframe/2, and use IBM's toolkit (included) as well. And you can fine-tune performance with High C/C++'s eight levels of optimization for ultimate customizability.

Take advantage of OS/2's multi-thread capability.

Use High C/C++ EasyThread™ to enable/disable thread-local storage for any section of code.

Create Dynamic Link Libraries with the flip of a switch.

Use High C/C++ EasyDLL™ to ensure future compatibility with current libraries.

Find and fix errors fast. High C/C++ provides 600+ user-formattable messages that pinpoint the line and column number where a problem occurred, and diagnose the problem.

In addition to High C/C++ for OS/2, MetaWare supplies high-performance 32-bit native and cross compilers to OEMs such as AMD, AT&T GIS (NCR), CenterLine, IBM and Intel. These OEMs depend on MetaWare for superior products and support—and so can you.

For more information about features and High C/C++ compilers for other platforms, call (408) 429-6382, or email: techsales@metaware.com.



MetaWare®

High C/C++ for OS/2, NT, DOS
Windows, AIX, UNIX SVR4, and Solaris

Circle Reader Service Number 2

MetaWare, the MetaWare logo, and High C are registered trademarks, and High C/C++, High C++, EasyDLL, and EasyThread are trademarks, of MetaWare Incorporated. All other company and product names are trademarks or registered trademarks of their respective companies. Copyright 1994 MetaWare Incorporated.



With so many advanced features in Micro Focus COBOL, there's one you may have overlooked.

The Future.

At Micro Focus, we have a history of ensuring a bright future for our customers.

Fifteen years ago, when the "future" was personal computers, we brought the first true business application development environment to the desktop. Over the years, we built on that foundation by providing the tools and utilities that delivered new levels of productivity to programmers.

Today, we're ready for the next step: Introducing Object COBOL™—the first true object oriented business programming environment.

The Micro Focus Object COBOL Option provides all the functionality you would expect from

an object-oriented development environment—including encapsulation, polymorphism and inheritance. It also brings all the benefits of object orientation—reusability, real-world modeling and increased maintainability—to COBOL programmers without discarding existing investments in code and skills.

Object COBOL is shipped with a library of classes for managing collections of objects and for creating Graphical User Interfaces. These classes can be accessed and extended with another Object COBOL component, the Browser.

Object COBOL even extends the COBOL language by letting you define syntax that best

suits your business needs. Object COBOL's unique vocabularies make applications easier to read, write and understand... for programmers and end-users alike!

And the best part? It's designed for COBOL programmers, so with Micro Focus, if you know COBOL, you're ready for an object-oriented future today.

For the latest update on this new technology, call 800-MF-COBOL and ask for our white paper: The Object Oriented COBOL Model.

Micro Focus: The past, present, and future of programming.

MICRO FOCUS®

Micro Focus is a registered trademark of Micro Focus Ltd. Object COBOL is a trademark of Micro Focus.

Circle Reader Service Number 3

os/2 Developer

NOVEMBER/DECEMBER 1994

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT



EDITOR'S COMMENTS

In Search of... 5



GUI CORNER

Just a Matter of Editing 6



PROGRAMMING INSIDER

Who's Up First? Thread Priorities and Scheduling 12



GRAPHICS

Color Palette Management with OS/2 20

By John D. Webb

OpenGL on OS/2 34

By Suzy Deffeyes and John Spitzer

Line Styles and the OS/2 PM GPI 54

By Nick Hodapp

A Peek at Taligent: The Graphics Subsystem 66

By Dean Roddey



OBJECT-ORIENTED PROGRAMMING

Integrating OpenDoc Parts with Workplace Shell Objects 29

By Scott Broussard and Eric Snell



DATABASE

Engineering Object Persistence with C-Set++ and DB2/2 46

By Liston Tatum



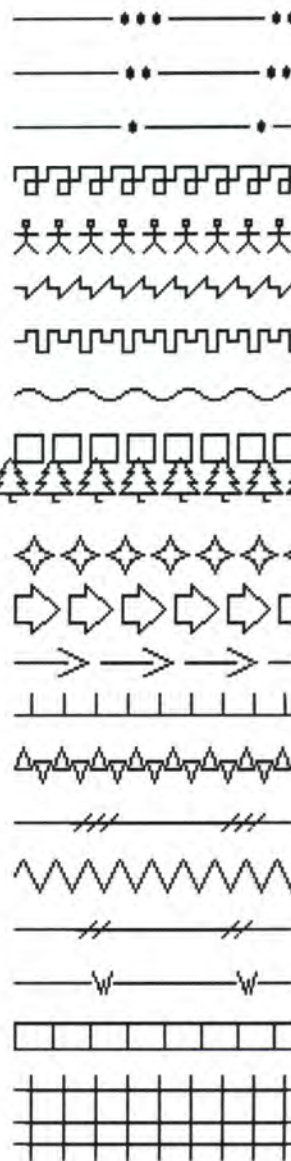
TOOLS

Graphical User Interface Tools Buyer's Guide 74



PRODUCT WATCH

New Products for OS/2 78





Programmer's Paradise®

Call for a
FREE
Catalog!



Watcom C/C++ V10.0 by Watcom

Comprehensive C and C++ development system for 32-bit DOS, Windows NT, Win 32s, OS/2 2.x, and Novell NLMS, and 16-bit DOS and Windows 3.x. Delivers productivity and performance, combining state-of-the-art compiler technology with a new integrated development environment (IDE) and comprehensive set of tools. Includes advanced GUI debugger, C++ class browser, profiler, and more.

Support for C++ templates, exception handling and Microsoft Foundation Class libraries (MFC).

List: \$199 Ours: \$189* FAX_{cetera} #: 1683-0012

* While supplies last.

key 01ND94

Watcom™ VX•REXX™ V2.1 by Watcom

Watcom VX•REXX Client/Server Edition is a visual development environment for OS/2. It includes all the features of VX•REXX V2.1 plus powerful connection, query, and chart objects that allow you to access several databases, manipulate data and chart the results quickly and easily. Features include: drag-and-drop programming; bound controls; professional multi-threaded, multi-windowed and drag-and-drop enabled application development; over 2 dozen objects; and much more.



**Special Introductory Price Only \$99
Client/Server Edition Only \$299
FAX_{cetera} #: 1683-0016**

key 03ND94



GpfREXX by Gpf Systems, Inc.

WYSIWYG visual OS/2 PM programming with REXX. Point and click to create PM design using the full CUA '91 control set (e.g., NoteBooks, Containers, etc.), as well as Multi-Media Controls. GpfREXX supports standard OS/2 features like Drag-and-Drop and Multi-Tasking as well as the extended features, including: Multi-Media, SQL (DB/2 including Remote services), and advanced communications (APPC, CPI-C, EHLLAPI). No royalties. (Compatible with Gpf 2.1)

**List: \$129 Ours: \$119
GpfREXX & GpfTools: List: \$274 Ours: \$253
FAX_{cetera} #: 3582-0005**

key 05ND94

HALO® Imaging Library™ for OS/2 by Lifeboat Publishing

HALO Imaging Library for OS/2 enables you to incorporate powerful image processing into your applications with over 100 C/C++ imaging functions. This advanced toolkit provides a full range of functionality including the ability to display, enhance, print, manipulate, and save most popular digital image formats. Special effects such as sharpen, blur, warp, color conversion, scaling, and more are also included.



**List: \$795 Special Price: \$599
FAX_{cetera} #: 1045-0016
Circle Reader Service Number 4**

key 07ND94



CA-REALIZER by Computer Associates

The first truly accessible approach to creating GUI applications. Using a structured superset of BASIC, you can quickly and easily create complete OS/2 and Windows applications without adding third party DLLs or learning about events, state-based systems, application cooperation, device control, and all of the other new techniques and complexities introduced by the other GUI programming systems. If you can create a program in DOS, you can create a program in OS/2 and Windows with CA-REALIZER!

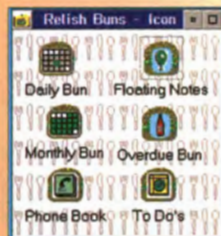
List: \$247 Ours: \$79* FAX_{cetera} #: 1004-0008

* While supplies last.

key 02ND94

Relish by Sundial Systems

Calendar, to do's, and phone book are fully integrated for total reliability, yet also independent for maximum flexibility. Schedule realistically for any time and duration, double-booking as necessary; categorize commitments; repeat events; print schedules; run programs; dial calls; prioritize to do's. Drag-drop, desktop objects ("Buns"), keyword searching, expert system for times/dates. Easy, convenient, CUA compliant, OS/2 Certified.



List: \$189 Ours: \$87 FAX_{cetera} #: 1015-9901

key 04ND94



Accusoft Image Format Library 5.0—OS/2 by Accusoft

Import, export, convert, compress, display, scan, image processing, and printing. 36 raster formats supported including: TIFF, PCX, GIF, BMP, TARGA, DIB, WMF, WPG, DCX, EPS, PICT, JPEG, Photo CD, Kofax, Cals, CLP, ICO, IMG, ASCII, IOCA, IFF, SUN, MAC, and MSP. Format compatibility guaranteed. Optimized for high performance. Image processing functions including: rotate, zoom, scroll, color-reduction, anti-aliasing and much more. Fastest imaging toolkit available.

List: \$1295 Ours: \$1243 FAX_{cetera} #: 2973-0001

key 06ND94

To order call: 800-445-7899
Corporate Developer Division: 800 441-1511
FAX: 908 389-9227
International: 908 389-9228
Customer Service: 908 389-9229
Programmer's Paradise Deutschland:
Tel.: 08121/79073
FAX: 08121/76566
Programmer's Paradise Italia:
Tel.: 39-2-967-00409
FAX: 39-2-967-02855



For more information on the products featured on these pages call
FAX_{cetera} #: (908) 389-8173

Programmer's Paradise

1163 Shrewsbury Avenue
Shrewsbury, NJ 07702-4321

* Prices subject to change without notice.
* Call for shipping charges/return policies.

9
8
7
5
4
4
0
0
8

EDITOR*Dick Conklin***EXECUTIVE EDITOR***Nicole Freeman***MANAGING EDITOR***Marta Dils***EDITORIAL ASSISTANT***Diane Anderson***EDITORIAL ARTIST***Peter Altenberg***VICE PRESIDENT/****GROUP DIRECTOR***Regina Starr Ridley***PUBLISHER***Cathy Passage***ADVERTISING SALES***Yvonne Labat**(415) 905-2353**Holly Meintzer**(212) 626-2275**Kristin Morgan**(212) 626-2498***MARKETING MANAGER***Susan McDonald***ART DIRECTOR/MARKETING***Christopher H. Clarke***MARKETING ASSISTANT***Larra Dutton***ADVERTISING PRODUCTION****COORDINATOR***Glenn Sadin***CIRCULATION DIRECTOR***John Rockwell***CIRCULATION MANAGER***Lucinda Formyduval***SUBSCRIPTION INQUIRIES***U.S.: (800) WANT-OS2**Foreign: (708) 647-5960***CHAIRMAN OF THE BOARD***Graham J.S. Wilson***PRESIDENT/CEO***Marshall W. Freeman***EXECUTIVE VICE PRESIDENT***Thomas L. Kemp***VICE PRESIDENT, PRODUCTION***Andy Mickus***VICE PRESIDENT, CIRCULATION***Jerry Okabe***BY DICK CONKLIN**

Editor's Comments

In Search of...

One of my favorite OS/2 features is the Information Presentation Facility, or IPF. This Hypertext-like program is used to create online help, tutorials, reference documentation, newsletters, and databases of all kinds. I've found several useful shareware IPF files (which have file names ending in .INF) that you may be interested in. I'll reference their CompuServe libraries, but you'll find most of them on other online systems and BBSs too.

Ever wanted a listing of OS/2 debugging tools or names of vendors offering C++ compilers? Search the online OS/2 Tools Guide. You can find it in CompuServe's OS2DF2 file library, section 14 (DAP), named OS2TG.INF. (To use the Tools Guide, just enter VIEW OS2TG.)

Can't find that article on visual REXX products in your back issues of OS/2 Developer? Look it up in the OS/2 periodicals database: OS2DF2, section 13 (OS/2 Developer Mag), file name MA94XX.ZIP, where "XX" is the latest monthly version. Thanks to Robert Simpson for putting in a lot of time and work on this one!

Mike Engelberg publishes an online newsletter called Developer Support News. It describes the latest offerings from IBM's Developer Assistance Program and other items of interest to OS/2 developers. Look in OS2DF2's section 14 (DAP), file name DSNXXX.ZIP. The "XXX" indicates the issue; "I" is an IPF version; "A" is ASCII. Larry Salomon's newsletter is called "OS/2 Electronic

Developer's Magazine" (EDM/2), and he uses IPF's graphics capability to add artwork such as screen dumps to his text. Look for it in OS2DF2, section 15 (Open Forum), file name EDMXXX.ZIP, where "XXX" indicates the issue number.

An online reference manual for IBM's C Set++ is called DDE4UL.INF. Look in OS2DF1, IBM C++ section (5).

Over in the OS/2 Support Library (OS2SUP), IBM Files (section 17) you'll find information on OS/2 2.1 performance tuning and other performance improvements in files 21TUNE.ZIP and 21PERF.ZIP. There are also several .INF files summarizing IBM's APARs (Authorized Program Analysis Reports) or bug fixes for OS/2.

You don't need to download a tutorial on OS/2's built-in REXX interpreter. Just look for C:\OS2\BOOK\REXX.INF on your hard disk.

That just scratches the surface. There are .INF files in most OS/2 libraries. If you don't include an IPF tutorial, help facility, or online reference with your application, it's worth looking into. You can find the IPF compiler on IBM's Developer Connection CD-ROM. OS/2's Information Presentation Facility can make life a lot easier for both developers and end users.

*Dick Conklin*

OS/2 Developer: Vol. 6, No. 6, November/December 1994

Subscription information: U.S.: One year (six issues), \$39.95. Canada, Mexico, and international surface mail, add \$16 per year for postage. Canadian GST no. 124513185. International air mail, add \$30 per year for postage. Foreign orders must be accompanied by payments in U.S. funds. For new orders and customer service, call (800) WANT-OS2 (800-926-8672) or (708) 647-5960 (fax: 708-647-5972). IBM employees and branch office customers can subscribe to OS/2 Developer through IBM Mechanicsburg's Systems Library Services (SLSS) using OS/2 Developer's order number: G362-0001. POSTMASTER: Please send address changes to OS/2 Developer, P.O. Box 1079, Skokie, IL 60076-8079. Allow eight weeks for subscriptions to begin. OS/2 Developer (ISSN 1073-0729) is published bimonthly by Miller Freeman Inc., 600 Harrison Street, San Francisco, CA 94107, (415) 905-2200. Application to mail at second-class postage rates is pending at San Francisco, CA, and additional mailing offices.

OS/2 Developer has applied for BPA membership.

© Copyright 1994 by Miller Freeman Inc. PRINTED IN THE U.S.A.

Miller Freeman, Inc.
A MEMBER OF THE UNITED NEWSPAPERS GROUP



*This is a continuation of our discussion of a 32-bit replacement list box. In this issue, we will show you how to add editing capabilities to the list box. By **MARK A. BENGE** and **MATT SMITH***

Just a Matter of Editing



Mark Benge

After having made some side trips lately in our journey, we will now use some of the ideas that we learned from our first side trip, which dealt with the entry field. In this installment, we will show how to integrate editing capabilities within the list box we have developed thus far.

WHERE TO BEGIN

Ah yes, that blank sheet of paper, also known as a tabula rasa. Don't you just hate it? Well, what we do know is what we want to happen. At least that is a start.

When we allow direct editing within the list box, we want to allow the users to use some keyboard/mouse combination to start the editing sequence. Then the users should be able to edit the text of the list box item with both the mouse and the keyboard. Once they are done editing, the list box should update the text internally to reflect the changes made by the users.

Okay, this seems easy enough, so what's the catch?

SMOKE AND MIRRORS TIME

Having laid out the operational rules in loose terms, we now come to the fun part (actually, that depends on your point of view) of implementing it.

Basically, from our perspective, the problem boils down to two or three items.

First, what is the mechanism that we will use to allow users to enter editing mode? This one is easy. When the users press the Alt key on the keyboard while pressing button 1 (the left mouse button usually) of the mouse over the list box item, we switch into edit mode. Second, what about the completion of editing? Here we watch for when the users' attention spans wane, that is, they move the focus elsewhere. When this happens, we deem that they are done and switch out of edit mode.

Great, we have nicely defined that interface, now on to the next problem, transparency. What is transparency? Basically, it is the appearance to users that they have switched from nonedit mode to edit mode without having any abnormal changes in the displayed appearance of the list box. Except for one condition (which we will touch on later), we don't want the text to bounce around or the background mix colors to be adversely affected.

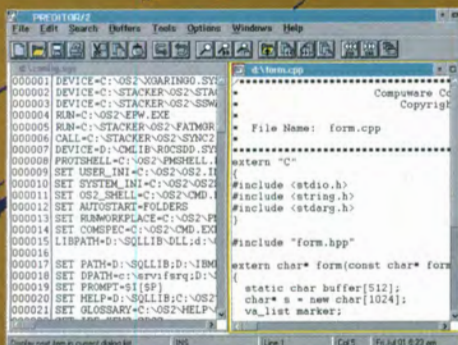
Where are the smoke and mirrors? They are in the final implementation. We have two ways of implementing the editing capabilities within the list box: a hard way (nope, don't like that, too



Matt Smith

For the untamed
programmer
only.

PREDITOR/2



Introducing PREDITOR/2.

Your source code has never fallen into the grips of a more powerful editor. PREDITOR/2 is for developers who don't have the time to deal with rigid programming environments, who must change facilities to fit their exact requirements and who need to dramatically extend editing functions to attain peak productivity. With this editor, you'll work with the speed and strength of a true predator.

HIGHLIGHTS:

- CUA-compliant GUI
- Multiple document interface
- BRIEF, VI, EMACS, ISPF, CUA emulation
- WorkFrame/2 integration
- Built-in compiler support
- Extensive customization facilities
- Powerful C-like extension language
- Unlimited Undo and Redo
- Powerful search and replace functions
- No limits on file sizes, buffers or windows

Special Introductory Price... \$149.00 • MSRP \$249.00

To order PREDITOR/2 or to receive a brochure, call now. 1-800-535-8707 or fax: 1-810-737-7119

PREDITOR/2 is a trademark of Compuware Corporation. All other company or product names are trademarks of their respective owners.
© 1994 Compuware Corporation.

Circle Reader Service Number 5



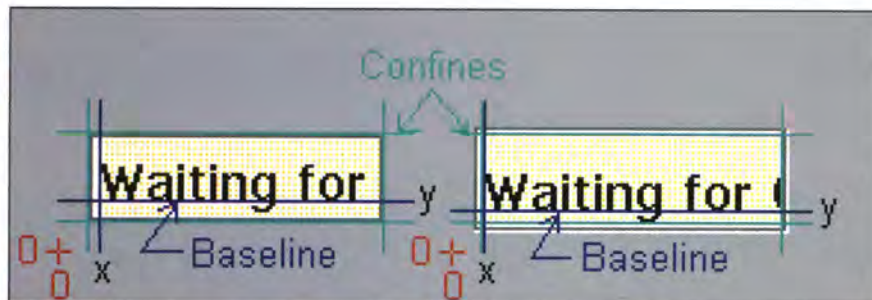


Figure 1. Entry field display design.

much work) and an easy way (that's more like it, you know we're lazy by nature!).

The hard way involves taking the code we developed for the entry field and merging it with the list box code. Don't wrinkle up your nose like that; it can be done, but as we said, it would involve a

lot of work. Mind you, everything would be seamless and fast!

The easy way has us using the entry field code we developed as a separate window. With a few adaptations to the entry field code, we have the editing of the list box item handled by our entry field.

Why our entry field and not the system's? Real simple: there are too many unknowns, and who knows when those kids in the Presentation Manager sandbox will change the mold to the entry field sand castle!

I THOUGHT I SAID...

The main problem of using the existing Presentation Manager entry field is that it does a few things we wouldn't really expect, like draw-

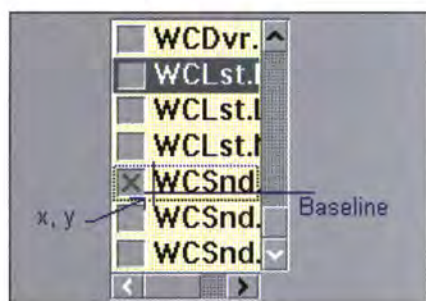


Figure 2. List box design.

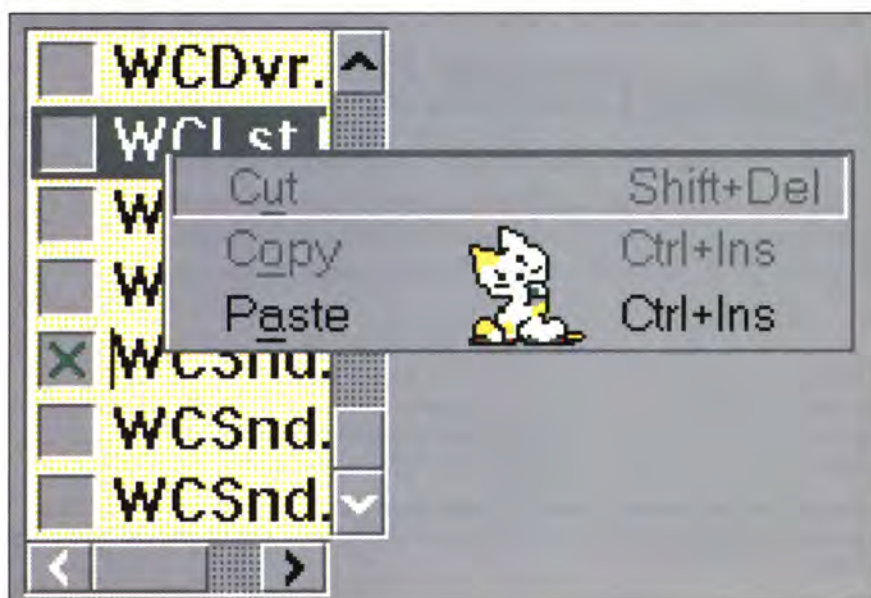


Figure 3. Pop-up menu.

ing outside its defined area. This aspect is how the border (ES_MARGIN) is drawn. Basically, the entry field outsets the border outside the specified limits and just draws them.

Then there is the problem of where the magic starting point is. The magic point is the location where the text starts relative to the origin of the control. The left entry field in Figure 1 shows the starting point (x-y) of our entry field for the drawing of the text. Please note that our entry field stays within its confines. The right entry field in Figure 1 shows the existing Presentation Manager entry field. Notice that the border resides outside the defined limits of the entry field.

Figure 2 shows the design criteria for the editing within the list box. Notice where the starting point is for the text. The x-y of the list box for the text should be the same for the entry field, thereby ensuring the transparency design criteria previously discussed.

A POINT OF CONTENTION

It is this starting point that is the crux of our problem. This is the starting point for the drawing of the text. What we need to be able to do is make sure that the entry field uses this same point for drawing the text. By doing so, we ensure that the text doesn't bop around and that the background doesn't seem to be having a bad hair day.

Since we don't know how the magic point is determined in the Presentation Manager entry field, we use our own entry field. We could try counting pixels within the Presentation Manager entry field, but like we said earlier, if they change their minds or make a mistake...

All we need to do to the entry field code to make sure it works properly in terms of the display of

PRESTO! A PRACTICAL CLIENT/SERVER SOLUTION!

Introducing GLS-Presto 2.0.

Now you can build state of the art OS/2 and Windows applications for hundreds and even thousands less.

Presto is the only client-server development tool which generates OS/2 and Windows applications in COBOL, C and C++, for one low price.

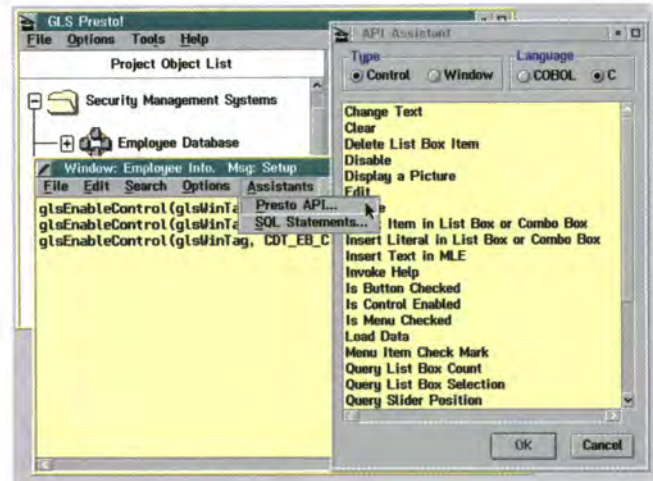
Presto is designed to help you become a client/server magician as quickly as possible without compromising on power, flexibility and security.

With Presto there is no need to invest in proprietary technology or suffer through an extensive learning curve. Presto allows you to take advantage of COBOL, C, C++ and SQL while effectively isolating you from the complexities of client/server development.



Presto's intuitive, easy to use integrated development environment will have you performing client/server tricks in no time at all. Design, prototype, generate, test and debug without ever leaving the IDE.

Presto makes the difference between OS/2 and Windows disappear, and lets you take advantage of the best of what they have to offer. For example, Windows MDI support is included for OS/2 and OS/2 control types may be used within Windows. Simply define your application, then select the target environment when you are ready to generate. Presto does the rest.



- Create complex applications in a minimum of time and run them under OS/2 and Windows.
- No knowledge of GUI or SQL programming required. The Presto code assistants are always ready to serve.
- Supports RAD methodology with instant proto-typing.
- Generated applications are quick and compact.
- Works with all major COBOL, C and C++ compilers.
- Incredibly fast. Applications generate in seconds.
- No compromises. MDI, tool bars, status bars, window geometry, custom controls and more.
- Use any DBMS or file access method compatible with your compiler.
- Fully networkable. No cumbersome object management required.
- Multi-media capability built in. Display GIF, TIF and Bitmap images dynamically.
- Absolutely no royalties or run-time fees.

Presto can appear on your computer for only



There is no better value.

To make some magic of your own, give us a call now at (219) 481-5809. Or, place an order via CompuServe, our mailbox is 73762,114.



the text is rework the function that calculates the display points for the entry field. In the `SizeEntryField` function, we toss out the calculations for the border, since we won't be showing one, and set the starting point for the text using the same calculations that would be used for the list box item.

The only difference is that we assume the x position of the entry field is the same x position of the text. This x position, from the point of view of the entry field, essentially gets translated to 0 in the end. The only thing that really needs to be done is to calculate the vertical position. Here we use the

same formula that is used to draw the list box item text.

Other than adding the code to the list box for the `WM_CONTROL` and the `EN_KILLFOCUS` and the `Alt+Button 1 down`, that is it.

Well, not quite. We added a new style `LSXS_EDITABLE` to denote that the entries within the list box are editable. This style is checked when the `Alt+Button 1 down` condition is detected. A position check routine is used to determine if the mouse is within a valid entry. When it is, the entry field is created using the rectangle position of the list box item.

The only other condition that

we had to handle was when the user clicked over a selected list box item. Here we had to play some tricks letting the entry field know what was going on so that it could draw the item in the selected state.

BONUS TIME

Since it took less time to add the support to the list box than we thought, we decided that we would give the CUA police something to grumble about.

If you click button 2 (usually the right mouse button) when you are in edit mode for a list box item, a pop-up menu will be displayed like that in Figure 3.

In the entry field code, we added the code to handle the button 2 down event. Here we check to see if we have previously loaded the menu, and if not, we load it from the DLL resources.

We then perform the checks on the clipboard to see if text is available for pasting. We also check to see if text within the entry field is selected, in which case we allow the copy and cut menu items to be enabled.

The second piece of code we added deals with the `WM_COMMAND` events for the entry field. When a valid menu item in the pop-up menu is clicked, we perform the appropriate cut, copy, or paste operation.

One trick here (no we aren't talking about the cat), which is not immediately noticeable but very important, is when the pop-up menu is displayed. As soon as the pop-up menu is displayed, the system places the focus on it. This causes the entry field to lose focus. Without change, the focus code for the entry field would then send a notification `EN_KILLFOCUS` message to the list box, which would then decide to toast the entry field. This is not what we want. Things go higgledy-piggledy and turn into a real mess.

Recursion 101 or Don't Try This Unless Accompanied by an Untrained Professional

The idea: when the `EN_KILLFOCUS` message is received by the list box, grab the text from the entry field, stuff the text back into the list box item, and then knock off the entry field through `WinDestroyWindow`. It works in theory, but in practice...

We implemented it exactly as outlined above. The last line before breaking from the case block was the `WinDestroyWindow` for the entry field. There was a slight problem though. We got another `EN_KILLFOCUS` sent to us. So, we tried killing the entry field again in the same code with the `WinDestroyWindow`.

Well folks, we got ourselves a fine case of recursion. Since we were in a `WinSendMsg` (this is how all notification messages are sent to the owner), the system hadn't had a chance to finish with the first `WM_SETFOCUS` message. So, in its humble opinion, it decided that it needed to remove the focus before toasting the control. In our infinite wisdom, we decided to be slick and write some compact code. Eventually, we would have run out of stack space, fallen down, and gone BOOM!

The solution, was to create a private message that we posted to ourselves using `WinPostMsg` (`WinSendMsg` wouldn't have helped since it is just like a function call to the window, and we would still have been within the processing of the notification code) so we would receive the message after most of the focus processing was complete, and then we could safely kill the entry field.

So what we do is check to see if the window gaining focus in the `WM_SETFOCUS` code is the menu and if this is the case, we simply bypass the notification code. This allows for the proper processing of the pop-up menu and for the entry field to receive the `WM_COMMAND` in an appropriate manner.

One final note, we suggest you take a look at the resource file. You will see a neat trick there that has never been properly documented since OS/2 Version 1.2. If you can't see it, drop us a note and we will let you in on the secret.

Mark Benge, IBM Software Solutions, Cary, N.C., is a staff programmer who joined IBM in 1989 and has worked on

various CUA '91 controls for OS/2 2.x. He works in IBM user interface class library development, where he was involved in the implementation of C++ classes for drag and drop in C Set++ 2.1. Benge has a B.S. in computer science from Western Carolina University. He can be reached on CompuServe at 73532,2063 and on Internet at banzai@vnet.ibm.com.

Matt Smith, Prominare Inc., Toronto, Ont., Canada, is lead architect for the Prominare Development System, an OS/2 2.x advanced GUI development environment. He has been actively involved with OS/2 since 1988 and cofounded Prominare in 1990. Smith has a degree in architecture from the University of Waterloo. He can be reached on Internet at msmith@interlog.com.

REFERENCES

Credits: Screen captures were done through OpenShutter from One UP Corp. and touched up through PaintBrush in Win-OS/2.

You can download the source code and driver application, titled `LSTBX4.EXE`, from these electronic sources:

CompuServe's OS2DF2 forum (OS/2 Developer Mag section).

File Area 11 on the IBM PCC BBS, (919) 517-0001.

OS/2 Tools section on the OS/2 BBS for IBM TALKLink customers.



Desktop Observatory 3.1

New version 3.1 features:

- SOM based Desktop backup and recovery.
- Centralized logging by user, name, server, or text string.
- On the fly creation of any class of object and all of its settings.
- Comprehensive on line and hard bound administration guides.

800-525-1650

complete
Workplace Shell
Control

SOM integrated OS/2 Desktop Management & Security

- Builds an instant Desktop anyway you want it and anywhere you want it in seconds.
- Apply restriction settings so objects cannot be moved, copied, deleted, renamed, or seen.
- Set up mobile Desktops with security or standardize a network with one secure Desktop.
- Unlimited objects and profiles can be loaded locally or from across any network OS.

- Password protect folders, programs, and restrict programs to specific files and directories.
- Centralized administration from any machine attached to your network.
- Boot protection protects against unauthorized access from a floppy diskette.
- Starts your own REXX or C utilities when folders or programs open or close. Great for backups, statistics, copying files from a server, or positioning windows.

317-279-5157
Pinnacle Technology
P.O. Box 128
Kirklin IN 46050

MICROSAFE™
Information Security Software

OS/2™ Warp, Version 3 Compatible

© Copyright 1992-94 Pinnacle Technology





This issue's Programming Insider looks at threads, how they are prioritized, and how they interact.

By David Reich

Who's Up First? Thread Priorities and Scheduling



David Reich

One of the biggest advantages of OS/2 applications over others is their ability to have multiple threads. Not only can OS/2 multitask applications by treating each as one or more executable units that can be quickly and preemptively switched in and out of a single processor, but applications can be designed to have threads within themselves. Therefore, pieces of applications can be multitasked with each other and the rest of the system, providing maximum use of the processor and maximum throughput to the user of these applications.

While threads are conceptually straightforward, I find many who understand the idea but still have difficulty with the implementation, usage, and handling of them. In this issue's Programming Insider, we'll take a look at what a thread really is, how the system handles them, and how priorities and synchronization come into play. We'll also look at the implications of threads in a multiprocessor system (since OS/2 is now able to take advantage of multiprocessor computer hardware).

Threads on a uniprocessor system (currently the most prevalent type of PC) run virtually concurrently. That is, the switching of threads in and out of the processor is so fast (and can be done at any moment without warning to the thread) that you can view the threads as

running concurrently. On the new multiprocessor systems, threads actually will be running simultaneously.

THREAD DEFINITION

A thread is the dispatchable, executable entity in OS/2. In contrast to a process, a thread does not own anything. The process is the unit of resource management, or ownership within OS/2, while the thread executes code and manipulates the resources owned by the process. Each process has one or more threads to manipulate its resources.

A thread is an execution instance. This execution instance consists of a register set, a stack, and a priority. It is also referred to as the thread's context. Memory, file handles, and everything else associated with running the program are resources owned by the process.

The first thread created in a process (the one created within `DosExecPgm`) has special properties that other subsequently created threads do not. For example, Thread 1 of a process receives all signals sent to the process, is used for all per-process DLL initialization during library and program loading, and is used for exit-list processing when the process terminates.

Other threads in a process are created with calls to `DosCreateThread`; they execute specific (sets of) functions, com-

Join the visual generation.



Introducing IBM VisualGen,[™] a powerful, visual programming solution that lets you rapidly develop both client and server applications for heterogeneous environments.

VisualGen includes visual construction of GUI client applications and a powerful 4GL for building remote and local server applications. With a single, integrated definition and test environment, including the strongest



test facility in the industry, you can evolve from prototype to production efficiently. VisualGen's unique development approach supports the full range of client/server models.

Client execution environments include both OS/2[®] and Windows.[™]

And VisualGen exploits the DB2[®] and CICS[™] families for data integration and industrial-strength, high-volume transaction processing.

Joining the visual generation is just a phone call away. To order VisualGen or to receive a free demonstration diskette, call **1 800 IBM-CALL, Dept. SA011.** In Canada, call **1 800 465-1234, ext. 492.**

**SOFTWARE FOR APPLICATION
PRODUCTIVITY.**



Introducing VisualGen. The new creative force in client/server programming.

Outside North America, call: (Austria) 0222.21145.2500, (Belgium) 02.2253333, (Denmark) 80304545 (France) 05.030303, (Germany) 0130.4567.111, (Italy) 1670.17001, (Netherlands) 030.384040, (Spain) 900.100400, (Sweden) 08.7934004, (Switzerland) 01.4366233, (UK) 081.5757700, (S. Africa) 27.11.2249.111, (Finland) 90.459.4176, (Norway) 66.999300, or contact your local IBM office. IBM, OS/2 and DB2 are registered trademarks and VisualGen, CICS and "The new creative force in client/server programming" are trademarks of International Business Machines Corporation. Windows is a trademark of Microsoft Corporation. ©1994 IBM Corp.

communicate with each other via memory or messages, and coordinate with each other with semaphores.

SCHEDULING AND PRIORITIES

The most frequently asked questions address how OS/2 schedules threads, when, how priorities fit in,

and how priorities get dynamically modified.

OS/2 schedules threads using a round-robin algorithm within priority levels, with dynamic priority modification. OS/2 has four priority classes: Idle, Regular, Fixed-High (sometimes called

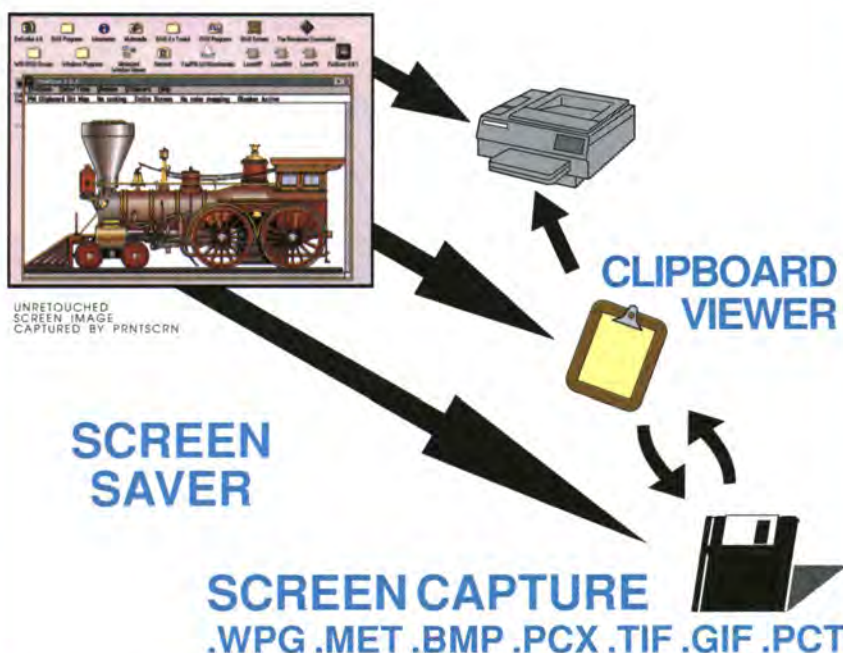
Server), and Time-Critical. As you might infer, Time-Critical is the highest priority and Idle is the lowest. Within each of these classes are 31 sublevels. The simplest way to view this is with Idle class having priorities 0-30, Regular class 31-61, and so on since thread priorities are all relative to one another when it comes to scheduling. Speaking of scheduling, the OS/2 scheduler is remarkably simple in its function. Of course how it does what it does is not quite simple, but to understand its effect on your threads, all you need is some common logic.

The underlying goal of the scheduler is to have the highest-priority ready-to-run thread be the one that is running in the processor. Others are in either a ready or a blocked state. On a uniprocessor system, only one thread is running at any given moment. If more than one ready-to-run thread has exactly the same priority, the system will run each in turn in a round-robin fashion.

A thread in the running state is, of course, the one running in the processor. Other threads are obviously not running, but some are also not ready to run. Something has blocked these threads. One reason a thread can be blocked is because it is waiting on a semaphore. A semaphore is a data structure used to coordinate threads either within or between processes that need to access a common resource. Semaphores will be discussed shortly.

Another reason a thread can be in the blocked state is for an input/output (I/O) request. When a request is made for I/O, such as a `DosRead` request, the thread traverses its way down into and through the file system and into the disk device driver (assuming the `DosRead` is for a disk file). Inside the disk device driver, the commands to the hardware are issued,

SINGLE KEY-STROKE SCREEN PRINTING



- Quality **Color or Black & White** copies of color **Desktop Images**.
- Copy any portion of any image on the Desktop.
- Screen Saver prevents monitor burn-in.
- Date & Time Display for Time-Stamping images.
- LAN installable. Complete support for LAN attached printers.
- Economical! 4 Utilities for 1 Price! Entity licensing available.
- The most stable and reliable product available.
- Includes both OS/2 2.1/2.0 (32-bit) and OS/2 1.3 (16-bit) versions.

THE LEADER: **PrntScrnm**TM Screen Image Utility for OS/2



TM

MITNOR Software

Phone: (918) 357-1628

Fax: (918) 357-2869

SCREEN IMAGES IN OS/2 DEVELOPER ARTICLES ARE CAPTURED USING PRNTSCRNM, COURTESY OF MITNOR SOFTWARE

Circle Reader Service Number 8

and the device driver executes a block instruction, forcing the thread into a blocked state. This effectively gives up the remainder of the thread's time slice and allows the scheduler to context switch in another thread to make efficient use of the processor. When the hardware responds to the request, the thread blocked in the device driver goes from the blocked state to the ready state. At that point, it is eligible to be given to the processor by the scheduler, based on its priority.

PRIORITY MODIFICATION

When a thread is first created, either implicitly from `DosExecPgm` or explicitly via `DosCreateThread`, its priority is in the regular class. Where within the class is not important since the priority of threads in the regular class is always being modified. Once a thread is created, its priority is modified by either OS/2 or the thread itself (or another thread within the process).

Threads can modify their own priority using the function call `DosSetPriority`. This API allows you to modify the priority of the thread itself, all threads within the process, or even all threads within the process and all threads in processes descended from the current one.

OS/2 will dynamically modify priorities of threads within the regular class based on whether the thread is part of the foreground process, becoming ready as a result of an I/O block, or becoming ready due to starvation. OS/2's dynamic priority variation will not touch the priority of threads in either the Idle, Fixed-High, or Time-Critical classes. Since priorities in the regular class are modified dynamically, it does not seem to be of much value to modify a priority with `DosSetPriority` to put the new priority in the regular class, unless of course you are moving a thread back into the regular class from one of the other three.

The dynamic priority modification algorithms are a complex part of the system. The mechanics are not as important to you as understanding their implications. For example, you should not have to worry about a thread of your application starving if it is in the regular

class or higher since the scheduler will dynamically modify regular-class threads based on starvation, raising the priority as time goes on to make sure it eventually gets some processor time. The higher-class threads eventually will get processor time by virtue of their



OnCmd® xBase for OS/2 ... A Winning Combination!

OnCmd harnesses the power and the speed of OS/2, enabling you to develop new applications or convert existing xBase applications, such as those developed in FoxPro®, Clipper®, and dBase®.



Preserving Your xBase Investment
OnCmd can quickly convert your xBase applications to run directly in OS/2's 32 bit Presentation Manager.



This minimizes development time and preserves your current xBase investment.

Text applications are automatically converted to a native Presentation Manager application that is both multi-user and network capable. With simple code changes, you can add buttons, check boxes, and drop-down menus to give your application more functionality.

Applications Development
OnCmd's screen painter, client-server facilities, "event driven" programming, and Dynamic Link Libraries (DLL) all add to the development power that you need.

A Performance Winner
OnCmd is a performance winner! No windows emulation overhead. Disk requirements less than 1 MB.

Installation under 5 minutes. In addition, it typically indexes large databases twice as fast as the competition in half the disk space. For more benchmark details, call or email us.



Client/Server Ready
OnCmd allows you to create a client server environment without the need for a dedicated server system. You can

evolve into the world of client/server at your own speed and at a very low cost, allowing you to take advantage of the improved performance and application stability of client/server.

*Go for the gold.
Order your
copy today!*

ON-LINE

ON-LINE DATA
5 Hill Street, P.O. Box 65, Kitchener, Ontario, Canada N2G 3X4
Tel: (519) 579-3930 Fax: (519) 579-2130
CompuServe: 70022,104 Internet: oncmd@onlinedata.com

On-Line Data recognizes the following trademarks: FoxPro (Microsoft Corporation), dBase (Borland International Inc.), Clipper (Computer Associates International Inc.), OnCmd (Online Data).

Circle Reader Service Number 9

priority. Additionally, Time-Critical class threads are guaranteed to be dispatched to run within 6 milliseconds of becoming ready.

FOREGROUND BOOST

A foreground boost is given to the threads belonging to the process in

the foreground, the one with which the user is currently interacting, to ensure responsiveness to the user. It is not necessary for an application to adjust its threads' priorities when going into the foreground; the system does it automatically. As you will also see, you

should change your own threads' priorities for only a few reasons—increasing responsiveness to the user is not one.

I/O BOOST

Another type of priority boost—and the one that gives developers the most trouble—is the I/O boost. An I/O boost is given to a thread when it becomes ready after an I/O block. In the previous disk device driver example, when the hardware is done and the thread is made ready, the priority is boosted. Upon return from the hardware I/O request, the thread is given an I/O boost not only to be nice, since the thread gave up its previous time slice, but also because it was in the processor just before the I/O request was made. At the time, it was the highest-priority thread in the system, and it is likely that the operation that caused I/O needs to be completed.

STRATEGIC PRIORITY MODIFICATION

The most frequent pitfall programmers encounter is the way threads of different priorities (not in the regular class) interact. One example I have seen is not unique and can be illustrated with a three-thread program. Thread 1 (1) is the user interface thread. Thread 2 (2) reads data from a device and fills buffers with this data. Thread 3 (3) empties those buffers and operates on the data.

Since 2 is an I/O-bound job, the programmer in the example puts its priority in the Fixed-High class, so whenever the buffers become empty, 2 would be up there in priority so it could be dispatched. The rationale is that because I/O devices are usually the slowest, performance could be improved by making sure the slowest operation could occur as often as possible. Generally there is nothing wrong with this approach. An I/O boost may not always be enough to



Supercharge Your REXX!

GAMMATECH™

REXX SuperSet/2

Supercharge your REXX with the most complete set of REXX external functions available - GammaTech REXX SuperSet/2.

With over 300 functions from seven DLLs, the GammaTech REXX SuperSet/2 will initiate File and System operations, issue Network commands, execute Video I/O, manipulate Processes and Semaphores, regulate the MacroSpace and perform Math calculations.

Call us today at (405) 947-8080 and you'll see what REXX can do with some extra power!

- Perform EXECIO functions
- Manage LAN Server users by group or domain
- Perform LAN Server permission functions
- Manipulate LAN Server files, sessions and shares
- Interface with NetBIOS, TCP/IP and Communications Manager/2
- Open, close, query and create semaphores
- Start a thread
- Destroy a process or thread
- Copy, move or rename individual or mass files
- Dump variable pools or MacroSpace to a file
- Load, drop or reorganize functions in MacroSpace
- Perform logarithmic and trigonometric calculations
- Query hardware components
- List current system environment variables
- Write character and attribute strings
- Obtain and set cursor type and screen mode



SofTouch Systems, Inc.
Workstation Division

1300 S Meridian, Suite 600, Oklahoma City, OK 73108
(405) 947-8080 • Fax (405) 632-6537

©GammaTech, Inc. 1991-94 All Rights Reserved

Circle Reader Service Number 10

ensure that 2 gets processor time when it becomes ready from an I/O to the device, so putting it up in Fixed-High should do the trick. It should also be noted that you should put it low within Fixed-High to make sure you don't interfere with other pieces of the system such as the shell.

The fallacy in this plan is when the developer puts 3's priority in Fixed-High as well. After all, it only follows that if the buffers are being filled at a Fixed-High priority, the buffers should be emptied at that priority as well. Right? Not quite.

Since 2 is an I/O-bound job, it blocks very often—every time a read request is sent to the I/O device, in fact. Thread 3 is a processor-bound job. If you look at each thread individually, things don't look so bad. Sure, you may slow down some other things by putting these threads in Fixed-High, but nothing too terrible should happen since they are at the bottom of that class. Looking at how these threads interact with each other may make you think twice, however.

Thread 2 goes to read data and then is blocked in the device driver. Thread 3 goes to empty buffers since it is now the highest thread waiting to run. Because it is processor-bound, it will run to the end of its time slice and not be blocked until the time slice is over or the buffers are empty. When 2 comes back from the I/O to the device, it sits waiting for 3 to finish since it gets no I/O boost in the Fixed-High class. When 3 is finished, 2 gets to put some of the data it read into the buffers. Now when it goes to read some more, 3 comes in and runs to the end of the time slice or until the buffers are empty. This continues on. You can see how these two threads at the same priority interact and how 3 is stealing time from 2 when 2 needs it more,

being the slower task.

To fix this, simply lower 3's priority by one notch, or even put it into the regular class. Now 2 will get the processor whenever it needs it, and 3 can still empty buffers (an inherently faster operation) efficiently.

As you can see, the primary concern in priority is how threads interact with each other, especially in relation to how they need to compete for resources. Priority is how OS/2 regulates the competition for the processor. One other item of note is to use Fixed-High



Smart-Lock

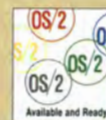
- a comprehensive security and configuration concept
- for host-connected OS/2 workstations
- based on Smart-Cards
- featuring the Chip-Controlled OS/2 Desktop



Smart-Lock is a security concept that controls the access to OS/2 systems by employing Smart-Card technology and a SOM-based restricted workplace shell.

Smart-Lock configures the OS/2 desktop according to the information on the E2PROM chip. Features include user specific boot protection and diskette access, single logon to LAN and host as well as the possibility to encrypt data and faxes.

Smart-Lock is CID enabled and allows easy installation and maintenance in large networks.



Distribution, installation and support in the U.S.A. and Canada through:



International, Inc.
621 NW 53rd Street, Suite 240
Boca Raton, FL 33487 USA
1-800-4SE-INTL or (407) 241-3428
Fax: (407) 278-3919

Circle Reader Service Number 11



and Time-Critical for specific uses such as data integrity. Do not use them to improve the performance of your application because it will be at the expense of overall system performance.

SYNCHRONIZATION AND PROTECTION

The other resources that threads can compete for besides the processor are memory and data. The OS/2 scheduler manages the contention for the processor, but it is up to you, the developer, to coordinate access to these other resources. You can synchronize access to resources in many ways through the use of flags and signals, but in a true multitasking system (and one that supports multiprocessors), you need system-supplied atomic operations to ensure integrity.

The tool provided by OS/2 is the semaphore. Semaphores are control structures that are designed to synchronize threads within and between processes where it is important that only one thread be accessing a resource at any given moment.

This discussion directly relates to the initial discussion outlining how processes own resources and threads manipulate those resources. All threads within a process have access to all of the process's resources, so it is important that if data is to be accessed by multiple threads, the access be tightly controlled.

OS/2 provides two basic types of semaphore: mutual exclusion semaphores and event synchronization semaphores. The former are designed to coordinate access to data, and the latter are designed to order or trigger events for threads to execute.

Just like thread priority, semaphore usage is relatively straightforward, but the implications can

be complicated. For example, if you have one thread holding a semaphore and requesting another while another thread is holding it but requesting the first, you have a deadlock. As the name implies, this is deadly to the application. This is especially important since there are many cases where your application is running in the context of a system thread. Deadlock in a Workplace Shell object, for example, requires drastic recovery methods.

Deadlocks can be avoided and resources safely coordinated by using classical computer science tools such as petri nets. The idea is to model sequences of access to the required structures as well as to the semaphores, since they too are being contended for. While this may seem an intimidating task, you should look at it. If it gets too complicated, you may be overthreading. If you can't feel comfortable about how your threads flow, you'll never be able to debug it. So either reduce the amount of threading or restructure it so you can understand it well.

Although there is no cookbook for designing threading and synchronization, the key to being "MP-safe" (for multiprocessor systems where you can have threads physically executing concurrently as opposed to fast context switching) is to take your tasks and break them down, designing threading in from the beginning. The way I usually recommend this be done is to draw black boxes for all of your tasks and look at which ones can be done in parallel and, more importantly, which ones cannot.

Once you determine which functions can and cannot execute in parallel, you can group them together to make multifunction threads. You can group functions that cannot execute in parallel to execute on one thread, ensuring

they can never possibly execute together. On other threads, you will have functions that cannot execute in parallel but can run in parallel with this first group. Your decisions will become a little more complex when you have some functions that you want to execute in parallel but that contend for resources. At that point, you will need to use mutex semaphores to coordinate access to the resources. You may even want to use event semaphores to order execution, but in general, you should weigh the benefit of running these functions in parallel vs. the cost of protection or the possibility of deadlock.

The worst mistake you can make is to take a single-threaded application and try to add threads to it. While you will have some level of success, you will never be able to gain all the benefits of threads without encountering many problems. You're better off redesigning from the beginning. In fact, if you do it right, you won't be writing code over again. You'll just be restructuring the functions to run them in thread pools, keeping those that cannot run in parallel in one thread, those that can run in parallel in another thread(s), and adding in coordination with semaphores. This is invariably cheaper and easier than if you had to debug all of the coordination problems you may introduce by adding threading in later.

David Reich has been with the IBM OS/2 development team since 1987. He has worked on many parts of the system, supported customers and application developers, and traveled the world giving seminars and teaching OS/2. He is currently working on the second edition of *Designing OS/2 Applications*, published by John Wiley & Sons. He can be reached on CompuServe at 76711,632 or via Internet at speedracer@vnet.ibm.com.

Don't even think about using a Relational Database System !



Use the POET Object Database for C++

C++ and class libraries have made the GUI development much easier. After all, it is only logical that more and more developers think in objects. But object orientation shouldn't end at the user interface programming level.

The Problem: Without POET, a C++ programmer must use flat files or a RDBMS to store objects. He has to write code to overcome the mismatch between the application and the database model. This leads to design restrictions, performance penalties and more code to write and maintain.

The Solution: POET operates at the object level, it speeds up the development process and provides greater performance. Furthermore, the developer can simply take the object-oriented application design in C++ and map it 1 to 1 into the database. Without any compromise at all!

Full featured database:

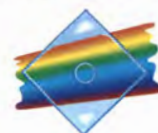
POET provides complete support for Encapsulation, Inheritance, Polymorphism and Containers as well as queries, object locking, and transaction handling.

True cross platform support: Complete interoperability makes the development of network enabled applications a breeze. POET supports Macintosh, Power Macintosh, Windows 3.1, Windows NT, Win32s, Windows for Workgroups, Novell Network (NLM), SUN, AIX, HP-UX, SGI, SCO, OS/2 and NeXTStep.

Comparing Object Oriented and Relational Design Methodologies

Feature	POET ODBMS	Relational RDBMS
Storing Objects	As Objects	Break Objects into Tables
Database Model	User Application Model	Separate Database Model Required
C++ Integration	Total	Poor
Database Operations	At Object Level	Must Write Code
Productivity	Increased	Reduced
Complex Object Performance	Excellent	Poor

Call 1-800-950-8845



POET
Software

POET Software Corporation, 4633 Old Ironsides Dr., Suite 110, Santa Clara, CA 95054, Tel.: (408) 970-4640 Fax: (408) 970-4630
United Kingdom (Silicon River) +44 81-316-7777, Germany (POET Software) +49 40-60-99-00, Australia (Microway) +61-3-580-1333, France (LCI) +33-1-34-65-7777
Netherlands (Protocols Software) +31 20 645 5023, Sweden (Duke Systems) +46 8 703 2781

Circle Reader Service Number 12



Graphics

*For graphic-intensive applications, OS/2's default color palette may provide insufficient color selection and control. These applications should use OS/2's color palette management APIs to enable the exacting color control they require. This article explores OS/2's color palette management system and the color control it provides. **By JOHN D. WEBB***

Color Palette Management with OS/2



John D. Webb

Color utilization is one of the most powerful aspects of graphics programming. The judicious use of color can communicate meaning and emotion, direct attention, emphasize detail, and add realism. To achieve these goals, precise color control is essential. On systems that provide 24-bit, "true color" hardware support, this isn't an issue since the 16 million available colors encompass the limits of average human color perception. However, because of the cost, both in hardware and performance, 24-bit color systems are the exception rather than the rule in the PC world today. Color palette management becomes the key issue with the far more modest, and far more prevalent, color systems limited to 256 colors or less.

In OS/2, the default 256-color palette provided by the GPI subsystem has been psychometrically tuned to provide a robust, generic spectrum suitable for most general applications. However, for color-critical tasks such as image processing, video postproduction, and virtual reality games, the default color palette may fall a bit short. Fortunately, OS/2 provides a set of color palette management APIs that allow applications complete control over the colors used for graphic drawing. Figures 1a

and 1b demonstrate some of the benefits of this API set. Figure 1a shows a 256-color bitmap that is displayed using the default color palette. Note the severe color banding in the sky and the harshness of details in the foothills. Figure 1b shows the same bitmap but uses the OS/2 color palette management APIs to create a custom color palette. Most of the severe banding is gone, and the details look more realistic.

A COLORFUL HISTORY

When Presentation Manager was first introduced with OS/2 1.1, application color control was provided by color tables. The fundamental principle of color tables was (and is) that there were two types: physical and logical. A single physical color table was associated with each output device (for example, a screen or printer) and contained the distinct RGB color definitions that a particular device directly supported. Logical color tables, on the other hand, were associated with presentation spaces (PS) and consisted of RGB color definitions the application used for drawing. Applications could create logical color tables that contained colors not directly supported by any of the output devices and that contained more color entries than could be simultaneously supported by

You Discovered the 32-Bit Power of OS/2.

FOR CORPORATE
DEVELOPERS BUILDING
SOLUTIONS ON
OS/2!

Now Master It at OS/2 DEVELOPMENT '95.



Introducing the conference for advanced software development.



he publishers of OS/2
Developer magazine have
assembled industry leaders to host the first
independent Pan-European conference on
OS/2. You'll sharpen your OS/2 skills
and learn from the experts about the latest
techniques and technologies to increase
your effectiveness on the job.

What's included:

- Excellent educational seminars
- Keynote speakers
- Product Roundtables
- And special events!



You'll Learn More About

- SOM programming
- Workplace shell programming
- User interface development
- OpenDOC
- Object-oriented programming
- REXX programming
- SmallTalk, and more!

Produced by

**OS/2
Developer**
magazine

May 8-10, 1995
Krasnapolsky Hotel
Amsterdam

To find out more about this
exciting OS/2 event, fax,
phone, write, or contact us by
electronic mail now!

Telephone:
32-2-538.60.40

CompuServe:
71673,1163

Internet:
CSP@mfi.com

Mail:
OS/2 Development '95
Chaussée de Charleroi 123a
Bte 5
B-1060 Brussels, Belgium

 **Miller Freeman, Inc.**
A MEMBER OF THE UNITED NEWSPAPERS GROUP



DEVELOPMENT '95

**SEND BACK THIS
COUPON TO
FAX 32-2-537.56.26**

Name

Title **Company**

Mailing Address:

Telephone

Fax **Email**



any device. When an application drew, its color choices were made from the logical color table. The system would then attempt to match the logical colors to physical colors either by “nearest-color” mapping into the physical color table for the target device or by dithering with colors from the physical color table. In this sense, a logical color table could be thought of as a “virtualized” physical color table, allowing an application to select and use color without concern for the actual color capabilities of each given device. In other words, color tables provided de-

vice-independent color usage for applications.

As nice as device independence was, however, there were times when absolute device-dependent color control was called for. So with OS/2 1.3, “realizable” logical color tables were introduced to extend OS/2’s color capabilities. On devices that supported this new feature, a logical color table could be realized into the physical color table; that is, an application could assign its custom color choices to the physical device’s hardware color palette. For the first time, applications had true

control over the actual colors used for their graphics. They were assured of accurate color representation without closest-color mapping or color dithering.

However realizable, color tables were not a perfect solution and had many shortcomings. The most glaring problem was that one application’s color selections were not protected from another application’s color selections. Even Presentation Manager’s desktop was vulnerable. For example, if application A realized a custom color table changing all the color slots in the screen’s physical color table, all applications running on the Desktop—and even the Desktop itself—would be drawn using application A’s colors. The realized color table would make application A’s graphics look great but could lead to horrendous-looking results for all other running applications. And even application A wouldn’t have it made because another application could realize a custom color table, “grabbing away” the screen’s color table from application A—with no notification! Because of this problem, it was strongly recommended that an application realize a color table only when it was the active window and its window was maximized.

The color palette management APIs were introduced with OS/2 2.0 to address the shortcomings of color tables. However, it wasn’t until the 32-bit Graphics Engine was released with the first Service Pack that device support for these APIs became a reality.

HOW COLOR PALETTE MANAGEMENT WORKS

Color palettes share many similarities with color tables. Like color tables, the fundamental principle of color palettes is that a logical color palette associated with a PS can be thought of as virtualizing the



Figure 1a. Bitmap with Default Palette.

physical color palette associated with an output device. Logical color palettes may be created and manipulated by applications for the purpose of creating nondefault colors. And logical color palettes may be realized into the physical color palette to dictate the actual colors used for output.

There are, however, some significant differences between color palettes and color tables, and this is where the true power lies. One major difference is that the system arbitrates among applications attempting to realize their logical color palettes. The arbitration follows a few simple rules. If an active focus window realizes its logical color palette, it will (with a few exceptions discussed later) be granted all of its requested entries in the physical color palette. If the window requests fewer entries than are available in the physical color palette, the remaining entries are marked as unclaimed. If a background window without the focus realizes its logical color palette, it will be granted only any currently unclaimed slots in the physical color palette. Note that all entries in the default color palette are considered unclaimed. When realizing a logical color palette, the system will resolve duplicate RGB definitions to a single color entry so the physical palette slots are conserved.

Figure 2 shows the relationship between two logical color palettes and a physical color palette after arbitration has taken place. App 1 has received all of its requested entries since it has the focus. App 2 has only two of its unique colors inserted into the physical palette, two slots that are unused by App 1. App 2's color entry 2 is mapped to physical slot 4, where the same color was realized by App 1. App 2's remaining two colors are mapped to the closest matching colors in the physical

palette. Note that although the index of each color in App 1's logical palette matches up with the index of that color in the physical palette, this is almost never the case. An application has no way to determine the physical palette index that a color has been mapped to, and it should never make an assumption about a physical index. Applications deal strictly with logical palette indices and must let the system handle the physical palette mapping.

Another significant difference between color palettes and color tables is that the system broadcasts

the message `WM_REALIZEPALETTE` to each window when palette arbitration is taking place. Arbitration is typically triggered by a palette-managed window gaining focus, or by an application realizing a newly created logical palette. This notification gives each application the opportunity to enter into the arbitration by realizing any logical color palettes it owns. It also allows the application to repaint its windows to take advantage of the new physical palette. It is important to note that the broadcasting of the `WM_REALIZEPALETTE` message occurs only if realization of logical



Figure 1B. Bitmap with Custom Palette.



color palettes cannot be resolved without changes to the physical color palette. In addition, the system protects against the broadcasted notification causing recursive loops and infinite palette arbitration.

With the improvements that color palettes provide over color tables, each window is assured of being displayed with the most accurate colors possible (assuming priority for the active/focus window).

HOW TO ENABLE COLOR PALETTE MANAGEMENT

This section looks at the steps involved in incorporating color palette management into an application.

Step 1: Determine If Color Palette Management Is Needed. Not every application needs to use color palette management. Overhead is involved in enabling color palette management, both in application coding effort and application execution. This must be balanced against the need for custom color control. For many applications, the

default color palette provided by Presentation Manager will do just fine. Or there may be other avenues to take, such as color dithering. When in doubt, a prudent approach might be to first code the application without using any of the color palette management APIs. If the results are acceptable, the overhead has been saved; if not, color palette management can usually be added without having to restructure the design of the application.

Step 2: Determine if Color Palette Management is Supported. Color palette management is driver dependent and is not available on all devices. Most notably, it is not available on VGA or true-color (that is, 16- or 24-bit color) devices. Support is most common on display devices supporting 8-bit color, such as SVGA, 8514A, and XGA.

An application can determine if color palette management is supported for a particular device by calling `DevQueryCaps()`. The bit flag

found at the defined location `CAPS_ADDITIONAL_GRAPHICS`, when used with the defined mask `CAPS_PALETTE_MANAGER`, will specify whether or not palettes are supported. A code example is shown in Figure 3.

Step 3: Create a "Logical" Color Palette. An application creates a color palette with `GpiCreatePalette()`. The logical color palette may have more or fewer color entries than are supported by the physical color palette. In fact, a well-behaved application uses as few color definitions as possible, leaving physical palette slots available for other applications.

Two options can be specified when the palette is created. One option, `LCOL_PURECOLOR`, designates that dithering is not to be used for any color that cannot be directly supported by the hardware. The other option, `LCOL_OVERRIDE_DEFAULT_COLORS`, designates that the entire physical color palette is to be used, even those slots that are normally reserved for system colors. By default, the Presentation Manager system withholds 20 physical color palette entries for use by system colors, allowing the system to maintain a consistent look in the user interface. Normally, these system colors are stored in the first 10 and last 10 physical palette entries. Without this option, an application is granted a maximum of 236 entries when realizing a logical palette. This override option should be used cautiously, as it can lead to an inconsistent look in the user interface. Note that the override option is in effect only when the window associated with the override palette has the focus.

The palette is defined by passing in an array of one or more RGB color definitions. The API prototype defines this parameter as an array of `LONG`, but an array of `RGB2` structures may also be used. By default, the flag byte of the `RGB2` definition is set to zero; however, several

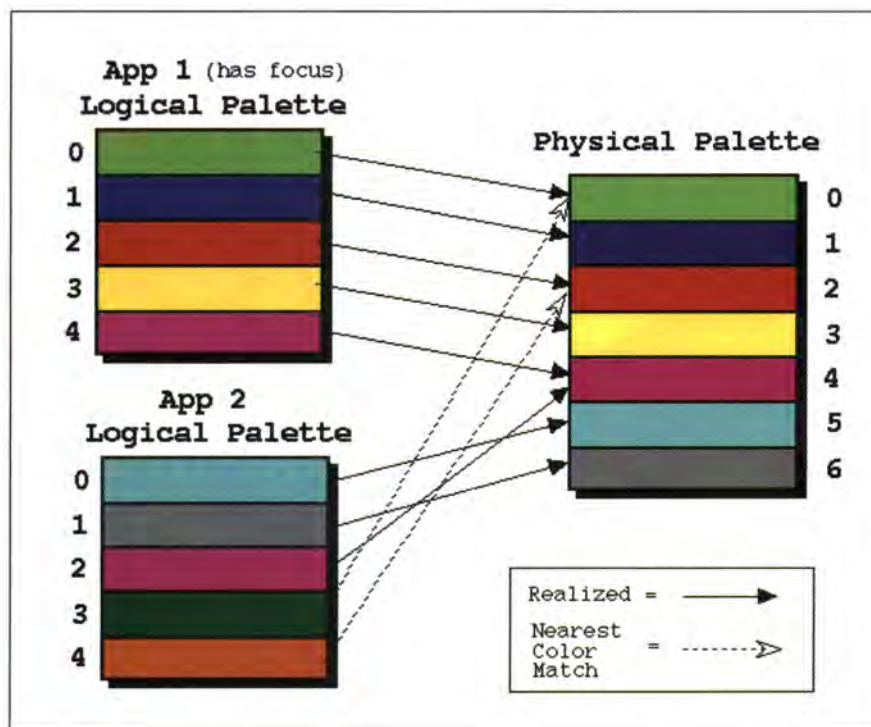


Figure 2. Palette relationship.



values can be assigned to this flag byte to affect the palette's behavior. Each RGB definition in the array can have a different flag byte value.

The value `PC_RESERVED` marks the entry as a reserved "animating" entry. Palette animation is accomplished with `GpiAnimatePalette()` allowing the application to rapidly change the RGB values of one or more `PC_RESERVED` physical colors. The changes are immediately visible on the screen and do *not* require window repainting. The other method of dynamically changing palette entries, `GpiSetPaletteEntries()` (which works for both animating and nonanimating slots), requires that the palette be realized and the window repainted before the changes are apparent. Although palette animation can create some dazzling effects, reserved slots should be used sparingly as they will not be shared with other applications' palettes; they can severely limit equitable palette utilization. Palette animation is hardware dependent and, therefore, may not be available on all devices.

The value `PC_EXPLICIT` is used to create a link to a particular physical color palette entry. This flag value indicates that the definition does not contain an RGB value but instead contains the index of a physical color palette entry. The index is stored in the `.bBlue` field of the `RGB2` structure (that is, the lower word of the `LONG`). This does not take a "snapshot" of the RGB definition in the specified physical entry. Instead, it maintains a link to that entry for the life of the palette using whatever RGB value is in that entry when referenced. The primary use of this value is for creating a logical color palette to monitor the physical color palette.

The value `PC_NOCOLLAPSE` is used to maintain duplicate RGB definitions when the logical color palette is realized. Normally, when a logi-

cal color palette containing duplicates is realized, the duplicate logical entries are mapped to a single physical palette entry. When `PC_NOCOLLAPSE` is specified, each duplicate logical entry is mapped to its own physical entry, creating duplicates in the physical color palette.

When a logical palette is realized, the color entries are mapped to available physical palette entries from the beginning of the RGB array. Since enough physical slots

may not be available to realize all the logical palette entries, the logical color palette should be ordered so that the most critical colors are placed at the front of its RGB array.

It is important to note that when the logical color palette is created, it is not associated with any PS and cannot be used. For the color palette to be useful, it must first be selected into a PS.

Step 4: Select the "Logical" Color Palette into a PS. A logical color

```
/*
 * Check to see if color palettes are supported. If so
 * then set the boolean bPaletteSupported to TRUE;
 * otherwise set it to false.
 */
/* ----- get screen DC to check palette support ----- */
hdcScreen = WinOpenWindowDC( HWND_DESKTOP );
/* ----- get capabilities bit flag from driver ----- */
DevQueryCaps( hdcScreen, CAPS_ADDITIONAL_GRAPHICS, 1, &lCapsBitField );
/* ----- check bit flag for palette support ----- */
bPaletteSupported =
((lCapsBitField & CAPS_PALETTE_MANAGER) == CAPS_PALETTE_MANAGER);
```

Figure 3. Determining if palettes are supported.

```
case WM_REALIZEPALETTE :
{
    ULONG ulColorsChanged ;
    /* ---- first check if palettes are supported ---- */
    /*
     * We will check our flag we set in Figure 3
     */
    if ( bPaletteSupported ){
        /* ----- realize the logical palette ----- */
        if ( WinRealizePalette( hwnd, hps, &ulColorsChanged ) > 0 ){
            /* --- colors were remapped, so repaint ---- */
            WinInvalidateRect( hwnd, (PRECTL)NULL, FALSE );
        }
    } // end of if ( bUsePaletteManager )
    else {
        return( WinDefWindowProc(hwnd, msg, mp1, mp2) );
    }
} // end of case WM_REALIZEPALETTE
break;
```

Figure 4. Handling `WM_REALIZEPALETTE` message.



```

/* ---- first, select default palette into HPS ---- */
GpiSelectPalette( hps, NULLHANDLE );
/* ----- now realize palette to trigger reset ---- */
/*
 * This is a kludge. For reasons unknown, it takes
 * three calls to WinRealizePalette() to reset
 * the default palette. Don't know if this will
 * change in the future
 */
WinRealizePalette( hwnd, hps, &ulChanged );
WinRealizePalette( hwnd, hps, &ulChanged );
WinRealizePalette( hwnd, hps, &ulChanged );

```

Figure 5. Resetting physical color palette.

palette is selected into (that is, associated with) a PS using `GpiSelectPalette()`. When a logical color palette is selected to a PS, it supercedes any previously selected palette or logical color table.

Calling `GpiSelectPalette()` with a NULL palette handle selects the system default color palette into the PS. Palettes may be selected into any type of PS: cached-micro, micro, or normal. However, to save

the overhead of selecting and realizing palettes into a cached-micro PS during every `WM_PAINT`, it's recommended to create a persistent micro or normal PS, selecting the palette into it once, and then use that PS for all drawing during the life of the application.

Palettes may be selected into PSs associated with any type of device context (DC) including `OD_MEMORY`, `OD_METAFILE`, and `OD_QUEUED`. A PS may have only one palette selected into it at a time; however, a single palette may be concurrently selected into many PSs. For example, the same palette should be selected to a screen PS and to a memory PS so color remains consistent when blitting between the two. When a palette is selected to a memory PS, it immediately and irreversibly changes the color data of any

TAKE YOUR TURN IN THE \$10 BILLION+ COMPUTER GAME INDUSTRY

Introducing . . . *Game Developer*, the magazine for programmers and developers of console, arcade, and home computer games. This publication is the information source for game programmers and developers looking for the latest trends in code, commerce and creativity.



GAME DEVELOPER

☐ **YES!** I want to be a charter subscriber to **GAME DEVELOPER!** Please send me a 6-issue subscription for only \$39.95.

☐ Bill me

☐ Charge my:

☐ VISA

☐ MC

☐ AMEX

Card # _____ Exp. _____

Signature _____

Name _____

Company _____

Address _____

City/State/Zip _____

Canadian/Mexican and International surface orders, add \$10 per year for postage. International airmail, add \$20 per year. Canadian GST #124513185.

OSD11

To Order: Copy this form and mail to Game Developer, P.O. Box 7703, San Francisco, CA 94120-7703 or FAX to 415 905-2562

bitmap selected into that PS. For this reason, it is usually best to select the palette before creating or selecting a bitmap into a memory PS. A palette selected into a metafile PS will be applied to the entire picture and must remain selected until the metafile is closed (that is, until the DC is disassociated from the PS). Palettes cannot be selected into PSs while creating or defining an area or path.

Step 5: Realize the Logical Color Palette. Once a logical color palette is selected into a screen PS, it should be realized with `WinRealizePalette()` before any drawing occurs. Only palettes selected into a screen PS need to be realized. Palette realization is not necessary for a PS associated with other types of devices, such as a memory PS or a printer PS. `WinRealize-`

`Palette()` returns the number of physical slots changed as an output parameter, as well as the number of colors that involved remapping physical indices as a return value. These numbers should be used only as indicators, not as hard, fast values, since continuing palette arbitration can invalidate them. As a rule of thumb, if the return value of `WinRealizePalette()` is greater than zero, the drawing area should be invalidated to trigger a repaint, thereby taking advantage of the new color mappings.

In addition to being called before using a newly selected palette, `WinRealizePalette()` should also be called whenever a `WM_REALIZEPALETTE` message is received. This system-generated message is sent to a window whenever it receives the focus, or whenever realization of a

palette by another application has triggered a round of palette arbitration. By calling `WinRealizePalette()` during arbitration, the application is assured that it will get the best color usage possible. Figure 4 shows an example of how to handle this message.

Finally, `WinRealizePalette()` must be called after any palette changes have been made with `GpiSetPaletteEntries()`. New color entries won't take effect until it is.

Step 6: Cleaning Up Color Palette Resources. When an application is shutting down, or has finished using its logical color palettes, it needs to perform a certain amount of palette resource cleanup. First, the application should get the handles of all selected palettes with `GpiQueryPalette()`. Next, the application should deselect the palettes





BACKUP, RESTORE, MANAGE AND SECURE YOUR WORKPLACE SHELL DESKTOP WITH **DeskMan/2**






**NEW!
VERSION 1.5
NOW AVAILABLE**

DeskMan/2™ is the first and only tool designed to manage the Workplace Shell™!

DeskMan/2 for OS/2 is unmatched in its ability to fully exploit the power and versatility of OS/2! Managers can use its security features to protect access to windows or objects. Administrators can easily and quickly standardize a department's desktops for maximum job efficiency. Individual users can quickly and easily migrate icons, objects and desktops between office and home machines. Or use **DeskMan/2** to backup and completely restore a custom desktop.

Incorporating the powerful **VUEMan/2™** virtual desktop & window manager; the potent new WPS snapshot facility; and more, **DeskMan/2** is packed with features!

New & Enhanced Features:

☑ Enhanced for ease of use ☑ Improved performance & functionality ☑ Improved window management	☑ Available object-level security ☑ Complete control over object menus ☑ New easy-to-use installer
--	--

Significant enhancements to every component!

☑ API & GUI for local & remote access ☑ CID enabled & LAD/2 supported ☑ Delete "undeletable" objects ☑ Prevent the deletion of any object ☑ Query any object for settings ☑ Secure individual windows ☑ Secure groups of windows	☑ Change multiple folders or icons all at once -- just drag and drop ☑ Full administrative control of features ☑ Restore, save, migrate any/all objects ☑ Total management of Workplace Shell ☑ And much more ... ☑ Call DevTech for current info!
--	---

DevTech
Development Technologies, Inc.
Software Development & Technology Transfer
308 Springwood Road, Forest Acres, SC 29206-2113 Voice: (803) 790-9230 Fax: (803) 738-0218

© Copyright 1994 Development Technologies, Inc. All rights reserved. DeskMan, DeskMan/2, VUEMan/2 and Workplace Shell are trademarks of Development Technologies, Inc. Workplace Shell is a trademark of International Business Machines Corporation. IBM and OS/2 are registered trademarks of International Business Machines Corporation.

Circle Reader Service Number 14



from the PSs by calling `GpiSelectPalette()` with `NULL` as the palette handle parameter. This will select the system default palette into the PS. Then the application must delete the palettes with `GpiDeletePalette()`. This is an extremely important step because palettes are considered global resources and are not automatically disposed of when the application closes. Undeleted palettes will remain in memory until the system is rebooted.

Finally, a well-behaved application should attempt to reset the physical color palette to remove any traces of its logical color palettes. There is no direct method to do this, but there is a *trick*. The application should call `WinRealizePalette()` three times in a row, using a screen PS with the system default color palette selected. This trick is not foolproof, but it should work in most cases. Figure 5 shows an example.

CONCLUSION

Whether the task at hand is playing a video presentation, displaying an MRI medical image, processing spectral satellite photos, or just trying to get the flesh tones correct on a favorite supermodel GIF, color palettes make the necessary color control possible.

And until 24-bit, true-color hardware is a standard on every system, OS/2's color palette management provides the best solution for equitable and economical sharing of limited color hardware on the multitasking desktop.

John D. Webb, Austin, Texas, is an independent consultant specializing in OS/2 development. He has worked with OS/2 since version 1.1 on projects both inside and outside of IBM. For the past two years, John has been with IBM's OS/2 Application Development Technical Support group in Austin. He holds a bachelor's degree in computer science from the University of Texas. John may be reached on CompuServe at 71075,1117.

Code: Source code for the sample bitmap viewing application used in Figures 1a and 1b will be available for download from CompuServe's OS2DF2 Forum, OS/2 Developer section. This loads the sample OS/2 bitmap file for viewing and can optionally use

palette manager or not. Download the file `VUEBMP.ZIP`.

Tools: The palette viewer also seen in Figures 1A and 1B will also be available from CompuServe's OS2DF2 Forum, OS/2 Developer section. Source code is not available, but the utility is free. The palette viewer, written by the author, monitors the hardware palette and shows all changes in real time. Individual entries, even animating ones, can be decomposed into their RGB components. The utility has an extensive help system. This is a useful utility for debugging palette management code. Download the file `PALVUE.ZIP`.

REFERENCES

OS/2 2.0 Technical Library Programming Guide Volume III (IBM Doc. S10G-6495).

OS/2 2.0 Technical Library Presentation Manager Programming Reference Volume I (IBM Doc. S10G-6264).

OS/2 2.0 Technical Library Presentation Manager Programming Reference Volume II (IBM Doc. S10G-6265).

OS/2 2.0 Technical Library Presentation Manager Programming Reference Volume III (IBM Doc. S10G-6272).

Winn, Graham C.E., *OS/2 Presentation Manager GPI: A Programming Guide to Text, Graphics, and Printing*, New York, N.Y.: Van Nostrand Reinhold, (ISBN 0-442-00739-6).



*Application functions can now be written in a document-centric way that allows their functions to be integrated into OpenDoc documents and the Workplace Shell. This article discusses some ways to share code and functionality between the two environments. By **SCOTT BROUSSARD** and **ERIC SNELL***

Integrating OpenDoc Parts with Workplace Shell Objects

With the advent of the Workplace Shell in OS/2 2.0, application developers were able to write object-oriented classes that define the behavior of data objects within the operating system's user interface and thereby extend the functionality of the operating system. The Workplace Shell supports many different types of objects and allows dissimilar objects to coexist within the same folder.

Now OpenDoc, a compound document technology on OS/2, will provide the ability to have data objects of different types coexist within the same document. Workplace Shell and OpenDoc have many similarities, but they also have some differences. As developers begin developing objects for either environment, they should consider what it would take to provide an object in both environments, in a well-integrated and interoperable way. This will give the user freedom to move the data back and forth between the Workplace Shell and the OpenDoc Shell and to view the data in a consistent manner.

Both environments can be extended through the use of System Object Model (SOM) classes, but each has a different set of methods and different object hierarchies. Each environment represents

objects as icons, and the icons can be opened into windows, although OpenDoc parts can also be represented within a facet or pane of the document. Both environments have storage mechanisms that allow the data for the object to be stored persistently; the difference being the Workplace Shell uses files and OpenDoc uses storage units, which are elements within a file. Although the programming environments are different in some key areas, object classes can still be structured to maximize the commonalities and reuse much of the code necessary to implement the desired function.

For example, if a text editing program is being rewritten as a Workplace Shell class and an OpenDoc part, the new code can be structured so that the core functionality is implemented independent of the environment. The ability to provide some environment-specific behavior can also be maintained if the class hierarchy is properly structured.

To share code between the Workplace Shell and an OpenDoc part, a third object would be created that contains the functionality necessary for the first two. An Editor object would define and provide the interface to the data-type object that is being edited. The Editor object would need to be factored into a data-type-specific subclass and an

environment-specific subclass.

Figure 1 shows a class hierarchy containing a data-type-specific editor, `TextEditor`, which inherits from `Editor` and environment-specific classes, and `Workplace Environment` and `OpenDoc Environment`, which also inherit from `Editor`.

In the figures, a line with a closed arrow indicates a class that inherits from another—an “is a” relationship. A line beginning with a circle indicates the construction of an instance of another object—a “has a” relationship. A line with an

open arrow indicates that one object is calling the method of another object.

Figure 2 shows an object hierarchy using SOM to implement an `OpenDoc` text part and a `Workplace Shell` data file object. The `Workplace Shell` class and the `OpenDoc` class would be written in SOM and would construct an instance of the C++ `Editor` class to use the core functionality. The `Editor` class contains methods the `Workplace Shell` object or the `OpenDoc` part needs to call to implement their functionality. These

methods fall into two categories: data-type specific and environment specific.

Data-type-specific methods deal with manipulating the underlying data and providing a view of that data to the user, while the environment-specific methods handle such things as where and how the data is stored and how the user manipulates the iconic data objects. This enables the environment in which the `TextEditor` is executing to be abstracted. The bulk of the code would be implemented as part of the `TextEditor` object.

If C++ directly supported dynamic multiple inheritance, this inheritance hierarchy might look a little different. The `Editor` class would inherit from `TextEditor` and either the `Workplace Environment` or the `OpenDoc Environment`. Although not supported directly, dynamic multiple inheritance can be simulated in C++.

In this scenario, the `Editor` class sits at the top of the hierarchy and defines all the methods necessary to manipulate both the environment and the data. The `TextEditor` class inherits from `Editor`, overrides all the data-specific methods, and defines those methods. On the other side of the hierarchy lie the `Environment`, `OpenDoc` and `Workplace` classes, and subclasses of `Editor`, which override and define environment-specific methods. When the user constructs an instance of `Editor`, `Editor` in turn constructs an instance of `TextEditor` and an instance of an `Environment` (either `Workplace` or `OpenDoc`). The `Editor`'s function is implemented through the use of the `TextEditor` object and the `Environment` Object.

When constructing the `TextEditor` or `Environment` object, `Editor` passes its object pointer to the new objects. This allows the `Environment` and the `TextEditor` to communicate through virtual func-

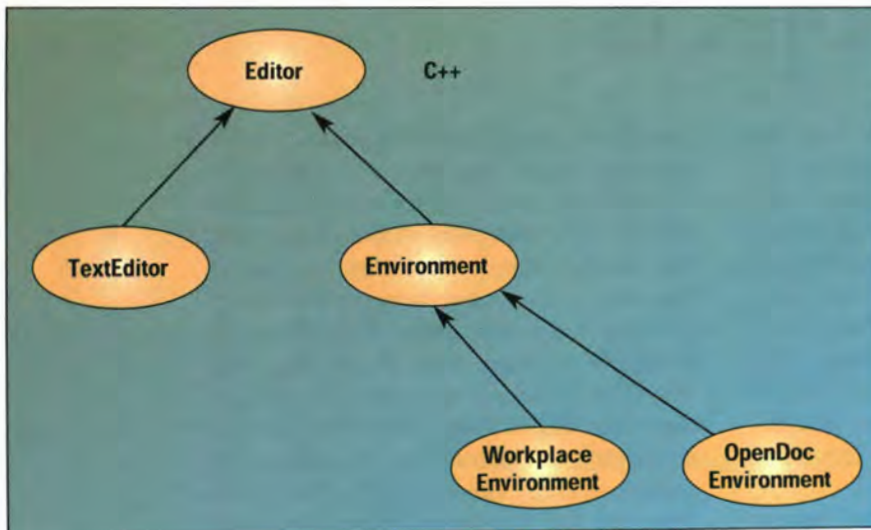


Figure 1. Inheritance hierarchy for a dual-environment `TextEditor`.

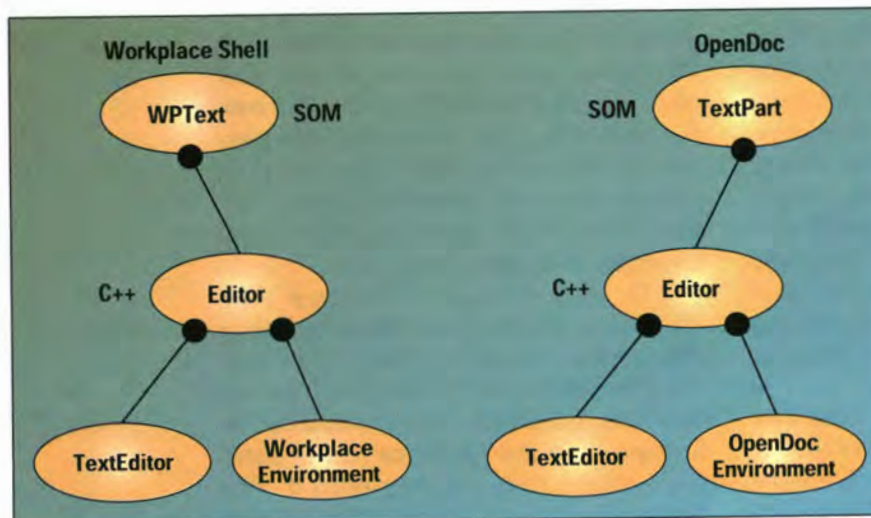


Figure 2. Object hierarchy in both environments.

tions. When the Environment class needs to manipulate the data, it calls back through the Editor object, which routes the request to the TextEditor. Likewise, when the TextEditor object needs to notify or manipulate the Environment, it calls back into the Editor, which routes the request to the specific environment object. The TextEditor and Environment classes inherit from the Editor object so they can implement the methods they override and call the Editor object for the ones they don't.

The Workplace Shell class, WPText, and the OpenDoc class, TextPart, when instantiated would construct an Editor object, which in turn would construct a TextEditor and either an OpenDoc Environment or a Workplace Environment. When WPText or TextPart calls the Editor class, the data-type-related methods are invoked on the TextEditor object, and the environment-related methods are invoked on the specific environment object. In other words, the user of the Editor sees one clean, simple interface, even though the functionality is implemented as three separate objects. This also avoids inadvertently invoking OpenDoc methods on Workplace Shell objects.

Dynamic multiple inheritance allows the functionality of the TextEditor to be shared between the WPText object and the TextPart object without having any specific knowledge of either environment. This sharing means not only less code for the implementors to write but also consistency for the user. When viewing and editing the text, the user is presented with the same windows and tools independent of the environment.

Some of the methods that would likely exist in the Editor class would be setTitle(), getTitle(), setIcon(), getIcon(), openEditWindow(), closeEditWindow(),

setObjectPointer(), getObjectPointer(), getObjectNamed(), setObjectNamed(), openData(), readData(), writeData(), and closeData().

Figure 3 shows an operational scenario that demonstrates how the method calls are routed to the appropriate object. The Workplace

Shell object, WPText, calls openEditWindow() in the Editor object that it created. The Editor object calls the openEditWindow() method on the TextEditor object that it created to do data-type-specific operations. The TextEditor opens the edit window, but to draw the con-



The secret's out!



- Native OS/2 spreadsheet •
- Objects to the core • REXX scripting •
- Small • Fast • Powerful •

MESA™ 2

for OS/2®



The spreadsheet that's more than the sum of its parts.

**TO ORDER: CALL
1.800.315.MESA**



17 St. Mary's Court
Boston, MA 02146
phone 1.617.734.6372
fax 1.617.734.1130



Circle Reader Service Number 15

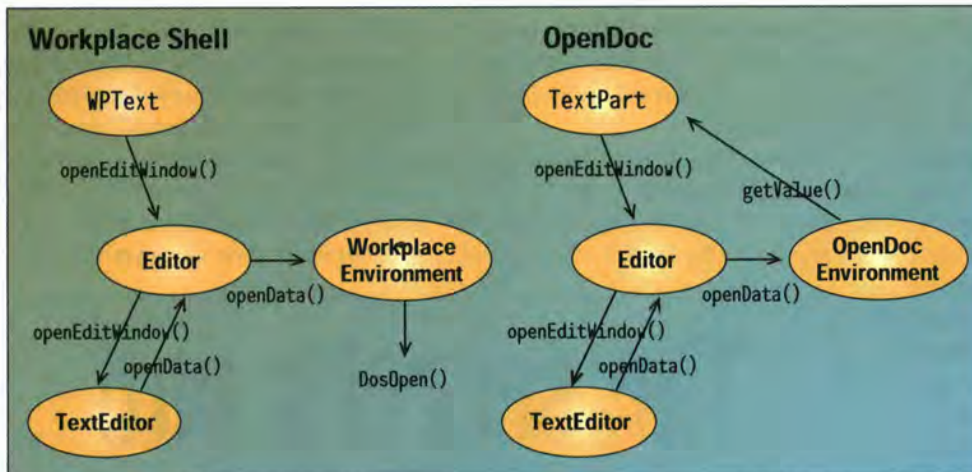


Figure 3. Operational scenario on both environments.

tent of the object, it needs to open the data container to access the text data.

The TextEditor calls `openData()` on the Editor object that created it. The Editor object in turn routes the

`openData()` call to the Workplace Environment object, which issues a `DosOpen()` on the file associated with the Workplace Shell WPText object.

The OpenDoc scenario is similar except that the OpenDoc

Environment object interprets `openData()` by calling `getValue()` on the storage unit associated with the OpenDoc TextPart. In this way, the environment-related operations, such as data storage, can be separated from the data-type-specific operations, such as drawing the text on the screen.

This demonstrates how a Workplace Shell object or OpenDoc part can share the same code to implement the Editor window of an object by structuring the class hierarchy to simulate dynamic multiple inheritance. OpenDoc parts also can be edited in place. This feature of a part has many functional similarities with the Editor window of the

A Perfect Fit Win-OS/2 Support with the Twain for OS/2 Toolkit



For more information
or
to order your developers kit call:

ST Solution Technology, Inc.
1101 S. Rogers Circle Bldg. 14
Boca Raton, FL 33487
(407) 241-3210 fax (407) 997-6518

Circle Reader Service Number 16

Read and Write dBASE files
from C, C++, and now REXX!

dbfLIB & dbfREXX



dbfREXX allows reading
and writing dBASE files
from your OS/2 REXX
programs!



dbfLIB includes support
for DOS, Windows,
OS/2 and Windows NT
all in one package!



Ready!
for OS/2

(713) 537-0318

Visa and MasterCard accepted.



dSOFT Development Inc.
4710 Innsbruck Drive
Houston, TX 77066

Circle Reader Service Number 17

part. This implies that the common text editing code should also be structured so it can be used by the standard editor frame window, which is constructed as part of TextEditor and can be used by the TextPart to implement in-place text editing.

When supporting both Workplace Shell and OpenDoc environments with common code, a natural extension to basic functionality is to allow conversion of an object from one environment to the other. The direct manipulation support in both environments affords an excellent framework to provide this functionality. Since most of the code to support the conversion is implemented in the shared code necessary for both environments, adding transparent drag-and-drop functionality is a relatively straightforward process.

Adding direct manipulation support for conversion between environments is accomplished by providing rendering mechanisms that allow each specific environment to accept data from another. The basic set of rendering mechanisms for transfer should be: render as a raw data buffer, render as a file, and render as an OpenDoc part.

A rendering mechanism for creating a file allows the user to drop an object into a Workplace Shell folder. The reason for rendering an OpenDoc object is obvious, but keep in mind that the object rendered must be containable by an OpenDoc container. Rendering a raw data buffer is a convenient mechanism for drag-and-drop between viewers and applications. Also, the code to render as a raw data buffer often can be used in many other places, including the two other mentioned rendering mechanisms. These three formats are sufficient, in most cases, to allow relatively simple movement of an object from one environment to another.

By using the object-oriented plumbing of the Workplace Shell and OpenDoc, you can maximize the power of both environments and allow users to manipulate data with an ease of use never before possible. Allowing the view windows of an object to be implemented by the same code within the OpenDoc environment and the Workplace Shell environment, the user sees a nearly identical interface and consistent level of function between the two environments. Also, supporting the movement of objects between environments presents a well-integrated system to the user.

For more information about OpenDoc you can order the De-

veloper Connection for OS/2 at 1-800-633-8266 within the U.S., or 1-800-494-3045 within Canada.

Scott Broussard is an advisory programmer in OS/2 Multimedia development. He is responsible for designing and developing multimedia user interfaces. Scott is currently the lead architect for the OS/2 Multimedia WPS/OpenDoc team. He can be reached via e-mail at sbroussard@vnet.ibm.com.

Eric Snell is an advisory programmer in OS/2 Multimedia development. His responsibilities include designing and implementing multimedia user interface and subsystem objects. Eric is currently the lead designer/developer for the OS/2 Multimedia WPS/OpenDoc team.



Take control of your OS/2 scheduling needs with
Advanced Task Scheduler for OS/2

September						
			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

ATS

Version 3.0

Are you looking for a way to manage your production job streams? Do you have programs that need to run after the completion of one or more other programs? Do you want to run programs on a regular basis? Do you want to process files that have just been received from another system or modified locally? Do you need to take action if a file is missing? Do you need to schedule around holidays and periods of heavy CPU load?

With ATS you can run your programs when YOU want them to run with out you having to be there.

Upgrades Available from Version 2 and other Task Schedulers
Call or E-Mail for Details

Here are some of the features of ATS for OS/2:

- * Manage Production Job Streams
- * Programs can be scheduled to run anytime.
- * Programs can be dependent upon Files, Other Programs, Holiday Schedules, and External Signals
- * Integrated Operators Console
- * Logging - To File and Console

New Features:

- * Job Queues - for managing your resources while managing your tasks
- * Periodic Scheduling
- * COMPLETE API and Command Line Interface

MHR
Software and Consulting
2227 U.S. Highway #1 * Suite 146
North Brunswick, NJ 08902
Voice: (908) 821 - 0359 * Fax: (908) 821- 0350 * CompuServe 70312,627

Only

\$349

Circle Reader Service Number 18



Graphics

OpenGL, a portable 3-D API, will soon be available for OS/2. This will bring to the PC market powerful graphics capabilities previously available only on workstations.

By SUZY DEFFEYES and JOHN SPITZER

OpenGL on OS/2



Suzy Deffeyes



John Spitzer

Have you ever wished you could use OS/2 to animate a car driving in fog at night? How about walking through a cathedral complete with tapestries and stained-glass windows? You'd have a lot of coding ahead of you if you planned on using graphical programming interface (GPI) calls alone. But imagine if you had some sort of graphical application programming interface (API) that would allow you to specify objects in three dimensions, give them their own material properties, possibly texture their surfaces, and render them at interactive speeds with lighting and many other options. Such an API has been defined by Silicon Graphics Inc. (SGI) and licensed to many different hardware and software vendors, including IBM, SGI, DEC, Intel, and Microsoft. This product, which is a trademark of SGI, is called the OpenGL API, and it will soon be commercially available for OS/2.

WHY OPENGL?

So what does OpenGL really get me as a developer? you ask. Actually, OpenGL provides a number of benefits to software developers. First of all, OpenGL is the first fully portable 3-D API. This means that the OpenGL calls you use on OS/2 will run identically on an OpenGL

implementation for the X Window System or Microsoft's Windows NT. This was achieved by separating the windowing and input handling functions from the rendering functions. As a result, window management and input must be handled by the native window system—Presentation Manager in the case of OS/2. You will also be assured to know that all implementations of OpenGL contain the same functionality. SGI requires an implementation to pass numerous conformance tests before it can use the OpenGL name. The OpenGL API also provides the developer with a wide assortment of 3-D rendering abilities.

Most importantly, the OpenGL API grants developers and users access to graphical hardware acceleration when it is available. Currently, many video chip and board manufacturers are intent on bringing OpenGL accelerators to the PC market as soon as possible. OpenGL accelerators already exist in the workstation market and are offered by companies such as IBM, SGI, DEC, and Evans and Sutherland. It is important to note that even in the absence of graphical hardware acceleration, today's newest processors (for example, IBM's PowerPC and Intel's Pentium) can deliver usable OpenGL performance—ac-



tually equivalent to or better than that of entry graphics workstations just five years ago.

HOW DOES IT WORK?

You may be curious about how OpenGL is implemented on OS/2 and Presentation Manager specifically. The OS/2 implementation is currently split into two DLLs: `OPENGL.DLL` and `RASTER.DLL`. `OPENGL.DLL` contains the utility library (GLU); the PGL calls (described in a following section); the OpenGL API entry points; and the graphics "pipeline," which performs object transformation, lighting, culling, and clipping. Any program using OpenGL must link with this DLL. This DLL in turn calls `RASTER.DLL`, which contains the graphics "rasterizer," which performs the actual drawing of the graphics primitives. Object data reaches the `RASTER.DLL` as vertices and is rasterized or turned into fragments—each of which represents an individual pixel in your window. These fragments then go through a series of possible steps including texture mapping, fogging, depth tests (Z-buffering), and blending. If a fragment is still around at the end of this process, it is put into an off-screen frame buffer (in the case of a pure software implementation) or directly into the video memory (assuming you have some hardware support). For the pure software implementation, a block transfer (blit) must occur to bring the image into video memory. This can be achieved in one of two ways: via DIVE (Direct Interface Video Extensions) or `GpiBitBlt`. DIVE allows direct access to the video memory by supplying the application (in this case, the `RASTER.DLL`) with a linear address to the memory and clipping information.

When ancillary buffers (depth, alpha, stencil, accumulation) and multiple frame buffers are not available on the video board, they are allocated in

main memory. Similarly, when rasterization acceleration is not supported on the video board, it is done in software. This is achieved by using a "pluggable" rasterizer, which resides in the `RASTER.DLL` previously mentioned. It is pluggable in the sense that those functions accelerated in hardware are "plugged in," while the rest is performed in software. This allows full OpenGL functionality and conformance in concert with as much graphics acceleration as possible. This rasterizer will be packaged as an OpenGL for the OS/2 Driver Development Kit (DDK) for use by hardware vendors supplying OpenGL accelerators.

INTEGRATION OF OPENGL RENDERING WITH PRESENTATION MANAGER

Because the rendering portion of the OpenGL API is intentionally windowing system independent, a portion of the OpenGL API must deal with windowing-specific issues. These calls make up what is known as PGL (Presentation Manager GL). The PGL calls will create and prepare an OpenGL context for rendering. This section will introduce you to these calls. Figure 4 depicts an OpenGL code sample that makes use of PGL calls.

The PGL interface provides two calls for specifying a visual configuration. A visual configuration is used when creating an OpenGL context. `pglChooseConfig(hab, attributeList)` will return a visual configuration that meets the minimum requirements based on the attribute list passed in. `pglQueryConfigs(hab)` will return a list of all available visual configurations, allowing you to choose the one that best suits your needs. Two PGL calls give details on OpenGL version and level of capability, `pglQueryVersion(hab)` and `pglQueryVersion(hab, pmajor, pminor)`.

You may have several different con-



A wide assortment of graphical primitives can be drawn

- Points, lines, segments, loops, triangles, quadrangles, triangle strips and more
- Utility library supports NURBS and complex polygons

Flexible data input

- Object data can be specified in 2, 3, or 4 dimensions and numerous data formats
- Allows developer to use immediate or display list mode

Complex light models

- Up to 8 lights
- Spot lights
- Local or infinite lights

Complex material models

- One or two sided
- Ambient, diffuse, and specular components can be set

Advanced texture mapping support

- Automatic generation of texture coordinates
- Two magnification filters
- Six minification filters including mipmapping
- Decal, modulation, or blending of texture

Line and Polygon anti-aliasing (removes "jaggies" or stair steps along edges)

Fogging effects

Accumulation buffer can facilitate motion blur and full scene anti-aliasing effects

Depth buffer and comparisons available for hidden surface removal

Alpha buffer and blending operations facilitate transparency and compositing effects

Stencil buffer can be used for Constructive Solid Geometry (CSG) modeling and shadows

Dithering allows true color images to be drawn with a small palette

Double-buffering allows for smooth animation

Table 1. OpenGL API features.

figurations to choose from. A visual configuration enumerates several different attributes. It can specify the depth and arrangement of the color frame buffer. The OpenGL rasterizer currently renders at 8 bits deep (3 bits red, 3 bits green, 2 bits blue) and at 24 bits deep (8 bits red, 8 bits green, 8 bits blue). Future versions will support 16 bits deep (5 bits red, 6 bits green, 5 bits blue).

A visual configuration will also specify what ancillary buffers are present. It can specify whether or not alpha, accumulation, stencil, or depth buffers are available. In the current software implementation, all of these buffers would be allocated in memory, which can be

expensive. For example, 64-bits-deep accumulation buffers allocated for a 400 by 400 window would allocate 1,280,000 bytes! Therefore, you should choose the visual configuration that requires the least number of buffers to suit your needs in order to minimize the number of buffers allocated.

Visual configurations are either single-buffered or double-buffered. Double buffering smoothly animates and renders. With double buffering, there are two frame buffers. The front buffer is displayed, and the back buffer is not. Each frame in the animation is rendered into the back buffer. When the full frame has been rendered, the back buffer is swapped

with the front buffer, thus displaying the newly rendered image. Single buffering is used when small updates to the frame buffer are made. Currently no single-buffered visual configurations are available, but they can be simulated by using a double-buffered visual configuration and calling `glDrawBuffer(GL_FRONT)`.

An OpenGL context is created using `pglCreateContext(hab, visualconfig, ShareList, IsDirect)`. In this call, `visualconfig` specifies the chosen visual configuration, and `ShareList` is a Boolean that determines whether display lists are shared between different OpenGL contexts in the same process. `IsDirect` determines whether an OpenGL context uses



You don't need weird hair to form a powerful peer group

Announcing LANtastic® for OS/2®

Guilty of going to extremes to win peer-to-peer connectivity for your team? Then consider LANtastic for OS/2, a true 32-bit multitasking, multithreaded peer-to-peer network operating system utilizing OS/2's superior power and performance.

LANtastic for OS/2 coexists with Novell® NetWare Requester™ for OS/2 and IBM® LAN Server clients and easily

communicates with LAN Server, Windows NT™ and other SMB-based servers. It has centralized network management and flexible security options that are easy to use.

Still need more proof? Call 1-800-809-1239 for the Artisoft Premier™ or Artisoft Advantage™ Partner nearest you. Then judge for yourself.



ARTISOFT®

Circle Reader Service Number 19

**See the proof
at Comdex
Booth #L4144**

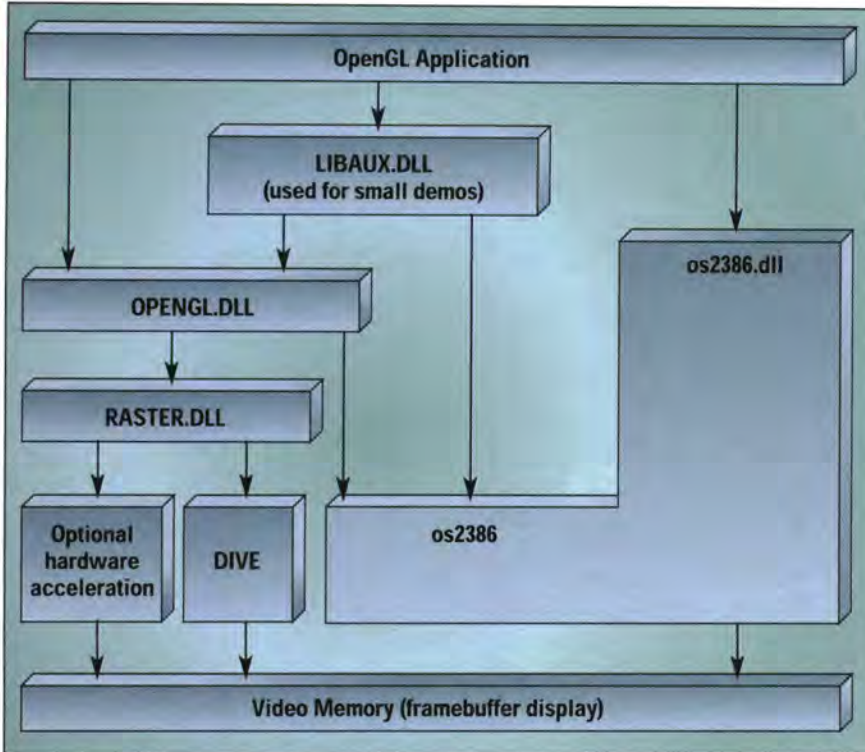


Figure 1. Typical OpenGL program structure.

the fastest means (`IsDirect = TRUE`) to display rendering or whether it uses `GpiBitBlt` (`IsDirect = FALSE`). After an OpenGL context is created, it must be bound to an OS/2 window before any rendering can occur. This is done through `pglMakeCurrent(hab, openglcontext, os2window)`. After this call, subsequent OpenGL rendering commands will use `openglcontext` to modify `os2window`.

An OpenGL context can be destroyed by calling `pglDestroyContext(hab, openglcontext)`. When this call is made, `openglcontext` must not be currently bound to a window. A context can be unbound from a window by calling `pglMakeCurrent(hab, NULL, None)`. If you want to copy a portion of the OpenGL state from one context to another, you can use `pglCopyContext(hab, src, dest, attributemask)`. In this call, `attributemask` specifies what portions of state are to be copied. If you wish to know if a context is a direct context or an

indirect one, use `pglIsIndirect(hab, openglcontext)`. The currently bound context can be queried using `pglGetCurrentContext(hab)`, and the currently bound window can be queried using `pglGetCurrentWindow(hab)`.

OpenGL rendering is not guaranteed to be visible on the screen until the graphics pipeline is flushed. This will happen whenever `pglSwapBuffers(hab, window)`, `glFinish()`, or `glFlush()` is called. It will also occur when `pglWaitGL(hab)` is called. If you wish to integrate OpenGL rendering calls and GPI rendering calls, you will need to have an indirect OpenGL context and make use of `pglWaitPM(hab)`, `pglGrabFrontBitmap(hab, phps, phbitmap)`, and `pglReleaseFrontBitmap(hab, phps, phbitmap)`. These calls will synchronize your OpenGL and GPI rendering, and they will allow you access to the presentation space handle and bitmap handle, which contain the OpenGL/GPI rendering.

OpenGL will also be able to make use of OS/2 bitmap fonts.

This will be implemented by putting the OS/2 font glyphs into OpenGL display lists. Currently support of bitmap fonts is planned through the use of `pglUsePMBitmapFont(hab, os2fontid, firstcharindex, count, listbase)`. For this call, `os2fontid` is the id of the OS/2 logical bitmap font; `firstcharindex` is the index of the first glyph; `count` is the number of glyphs; and `listbase` is the first display list to be created.

WINDOWING TOOLKITS

The OpenGL API intentionally does not include window creation, colormap creation, event and input handling, or other functionality that tends to vary between windowing systems. Therefore, you must use the functionality in the OS/2 Win and GPI calls to implement these functions when needed. There exists a need for simple common windowing and event functionality across platforms. This functionality exists in the form of several toolkits. These toolkits are provided "as is" with no formal support and are intended for simple demo development. They are available on several different OpenGL platforms. By recompiling, simple demo programs can run on all of these platforms.

The TK toolkit was the first toolkit offered when OpenGL was first implemented on the X Window System. It provides a simple interface. A few calls will create a window with an OpenGL context bound to it. Primitive keyboard and mouse input handling is also provided. The AUX toolkit runs on top of TK and offers similar functionality. This toolkit is used in the Addison-Wesley OpenGL Programming Guide samples.

RESTRICTIONS AND LIMITATIONS

There are some restrictions on the current version of OpenGL that you should keep in mind. First off,

Fast Visual Application Development for OS/2 and DB2



If you're looking for fast and easy application development for OS/2, then take a look at the award-winning Watcom VX•REXX visual development environment. VX•REXX lets you build applications to exploit the graphical user interface, multi-threading, and multi-processing power of OS/2. VX•REXX Client/Server Edition gives you the added power to access DB2 or other database systems, manipulate the

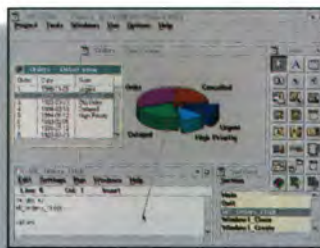
data, and chart the results at lightning speed.

"We like VX•REXX. Using it for development feels like driving a Porsche: it's fast, it's compact, everything's in the right place, and it makes us look good, too."
Peter Coffee, PC WEEK

Designed to Meet Your Needs.

Watcom VX•REXX combines a project management facility, visual designer and an interactive debugger to deliver a highly productive visual development environment. The Client/Server Edition includes additional powerful objects so you can rapidly create rich GUI database applications. You can create OS/2 client applications which connect to DB2/2 or DB2/6000. Use IBM's DRDA support on OS/2 to access DB2 for MVS, DB2/400 for AS/400, and DB2/VSE and VM (SQL/DS) for VM and VSE. Also supported are Watcom SQL and ODBC-enabled databases¹.

"Overall, this edition of VX•REXX for OS/2 is an outstanding visual client/server development platform." **Nicholas Petreley, InfoWorld**



Point. Click. And Presto!

To create an application you draw user interface objects, customize their properties using standard OS/2 notebooks, and define their event code using powerful drag-and-drop programming. To add database access just draw a query object, visually design a SQL query, press OK and presto— your window is automatically populated with objects that are bound to your query to display, update and search your data.

"Drag-and-drop nirvana." **Nicholas Petreley, InfoWorld**

Give Your Data a Whole New Image.

Energize your applications by displaying your data in a 3D chart. The Client/Server Edition gives you more than a dozen chart types to choose from, along with over 150 display options. You also get complete support for run-time events so you can bring new drama to your data by making your chart interactive.

"VX•REXX is a must buy." **Jacques Surveyer, ComputerWorld**

Standard or Client/Server Edition— Which one is for you?

To start creating powerful OS/2 GUI applications right away, order your copy of Watcom VX•REXX Standard Edition for just...**\$99***

Or, to start creating rich client/server database applications, order Watcom VX•REXX Client/Server Edition for just...**\$299***

- Over 2 dozen objects, including CUA'91 containers, notebooks, pop-up menus and more
- Integration and control of existing applications through DDE, keystokes or REXX API's
- Easy to learn event-driven programming model with complete on-line documentation
- Support for professional multi-threaded, multi-windowed and drag-and-drop enabled applications
- Code reusability through section and file sharing
- Graphically create CUA'91 Presentation Manager objects, quickly customize their properties, and easily attach REXX procedures
- Package your application as an EXE or PM macro for royalty-free distribution



1-800-265-4555

Watcom
A Powersoft Company

Watcom International 415 Phillip Street, Waterloo, Ontario, Canada N2L 3X2 Tel. (519) 886-3700 Fax (519) 747-4971 *Prices and specifications are subject to change without notice. Price does not include freight and taxes where applicable. Prices quoted in US dollars.
¹ ODBC drivers are available from INTERSOLV, Inc. Watcom, the Lightning Device, and VX•REXX are trademarks of Watcom International Corporation. Other trademarks are properties of their respective owners. ©Copyright 1994 Watcom International Corporation.

Circle Reader Service Number 20

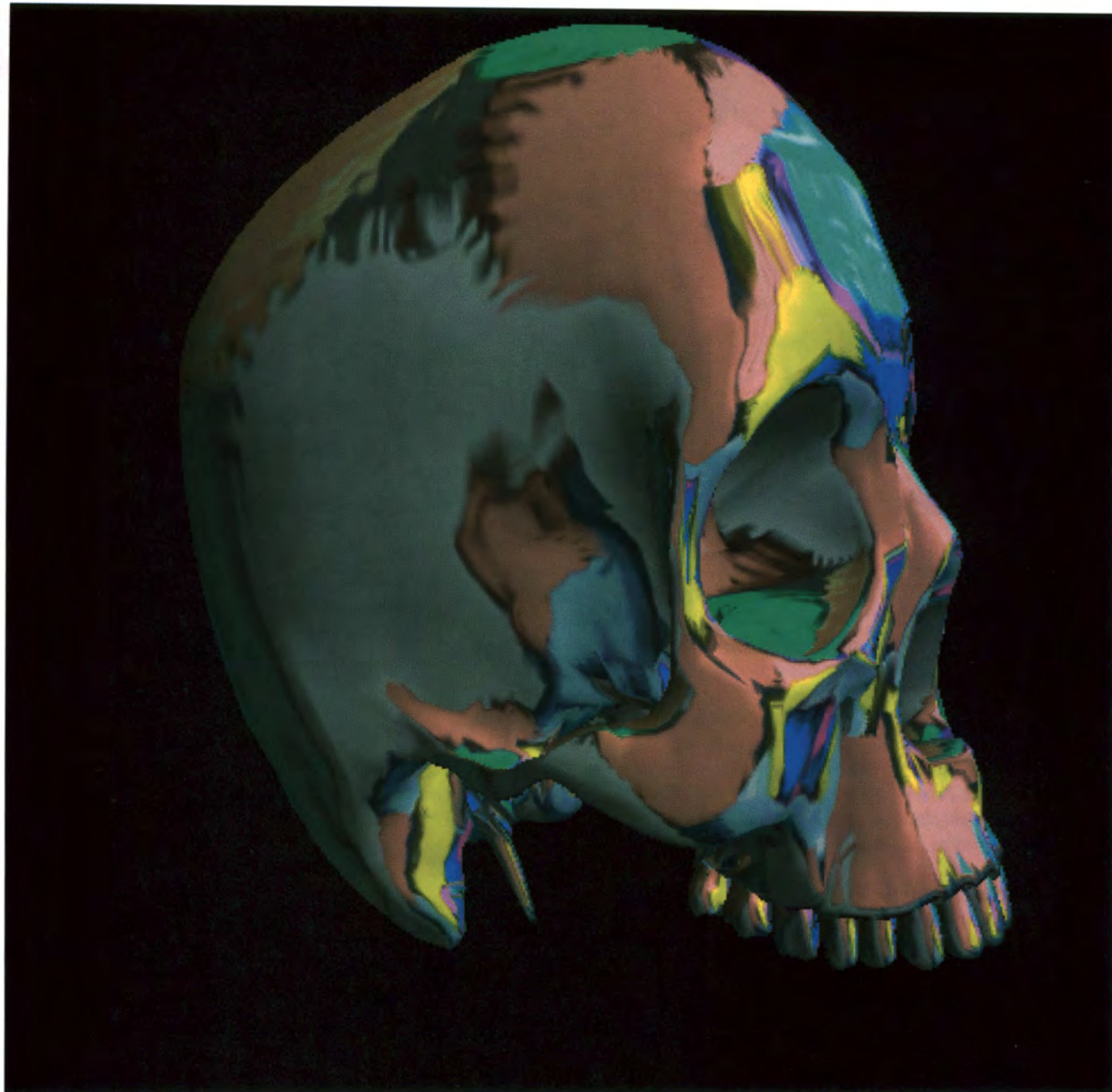


Figure 2. An OpenGL texture mapped image. Datasets provided by Viewpoint.

the OpenGL libraries are not thread safe, as the tradeoff in performance was too great. You should structure your programming in such a way that you have one OpenGL rendering thread and other threads that handle tasks such as message handling.

Also, you can have only one current context and one current window at a time. Additionally,

contexts and windows cannot be shared across processes. If you create a context in one process and try to bind it to a window in another process, you will get an error.

Windows that are bound to OpenGL contexts must be created with a client window class (`WinRegisterClass`) that includes class style type of `CS_SIZEREDRAW` and `CS_MOVENOTIFY`. After a window is

bound to a context via `pglMakeCurrent`, you should not call `WinSubclassWindow`. You can, however, call `WinSubclassWindow` before you bind to the window.

CURRENT OPENGL OFFERINGS

IBM has not announced a release date for OpenGL on OS/2. It is currently in beta, and the beta release is available on the OS/2



Figure 3. An OpenGL texture mapped image.

Developer Connection CDs and also by sending e-mail to gl-beta@innerdoor.austin.ibm.com.

CONCLUSION

The OpenGL API will grant developers and users of OS/2 access to graphics capabilities previously available only on expensive workstations. Its portability across numerous windowing systems, operating systems, and hardware platforms will allow application developers to spend less time porting their code and more time enhancing it.

Suzy Deffeyes is the team lead for the OpenGL on OS/2 project. She designed and coded the PGL specification that integrates OpenGL into the OS/2 environment. She also developed code for OpenGL in the X Windows environment and has experience in OS/2 business graphics application programming. She has a bachelor's degree in computer science from the University of

Texas at Austin. She can be reached via e-mail at suzyq@austin.ibm.com.

John Spitzer is a member of the Technical Staff at Silicon Graphics Computer Systems. John is the author of *GLperf*, a tool for measuring OpenGL per-

formance, which has been adopted as an industry standard benchmark by the OpenGL Performance Characterization committee. John received bachelor's and master's degrees in computer science from Rice University. He can be reached via e-mail at spit@sgi.com.

REFERENCES

OpenGL Programming Guide, Addison-Wesley ISBN: 0-201-63274-8, also available through IBM PUBORDER (SR28-5126-00).

OpenGL Reference guide, Addison-Wesley ISBN: 0-201-63276-4, also available through IBM PUBORDER (SR28-5125-00).

OpenGL Specification available using anonymous ftp from sgi.com in `/sgi/opengl/doc` PGL specification available on OpenGL beta and Developer's Connection CD USENET news group: comp.graphics.opengl.

OpenGL sample program detailing PGL calls is available on CompuServe: `OpenGL.zip` on OS2DF2 forum in the OS/2 Developer magazine section.



```
/* THE TEST CASE PROVIDED UNDER THIS AGREEMENT IS PROVIDED */
/* ON AN "AS IS" BASIS WITHOUT ANY WARENTIES EXPRESS OR */
/* IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARENTIES */
/* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE */

/* sample code for using OpenGL and PGL */
/* Draws a gouraud shaded octahedron */

#include <stdio.h>
#include "pgl.h" /* PGL calls */
#include "gl.h" /* OpenGL calls */

#define PM_ESCAPE 0x0f
#define MSGBOXID 22
#define SQRT2 1.414

/* attributes passed into pglChooseConfig */
int attriblist[] = {
    PGL_DOUBLEBUFFER, /* request doublebuffered visual config */
    PGL_RGBA, /* request rgb (true color) visual config */
    None /* always end list with this */
};

HAB hab;

void DispError(PSZ errstr)
{
    char buffer[256];
    sprintf(buffer, "Error (0x%x) in SAMPOGL.EXE:", WinGetLastError(hab));
    WinMessageBox(HWND_DESKTOP,HWND_DESKTOP,errstr,buffer, MSGBOXID,MB_MOVEABLE|MB_CUACRITICAL|MB_CANCEL);
    exit(0);
}

void Setup()
{
    glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_FALSE);
    glDepthMask(GL_FALSE);
    glEnable(GL_CULL_FACE);
}

float verts[][3] = {
    { 0.0, 0.0, (1.0/SQRT2)},
    { 0.5, 0.5, 0.0},
    {-0.5, 0.5, 0.0},
    {-0.5,-0.5, 0.0},
    { 0.5,-0.5, 0.0},
    { 0.0, 0.0, -(1.0/SQRT2)}
};

float colors[][3] = {
    {1.0, 1.0, 1.0},
    {1.0, 0.0, 0.0},
    {0.0, 1.0, 0.0},
    {0.0, 0.0, 1.0},
    {1.0, 0.0, 1.0},
    {0.0, 1.0, 1.0},
};

MRESULT EXPENTRY WindowProc(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2)
{

```

Figure 4. A simple OpenGL program using PGL interface (continued on page 43).



```
static float t = 0.0;
static SWP clientsize;
static USHORT mycode;
static UCHAR key;

switch(msg) {
case WM_SIZE:
    /* Upon a resize, query new window size and set OpenGL viewport */
    WinQueryWindowPos(hwnd,&clientsize);
    glViewport(0, 0, clientsize.cx, clientsize.cy);
    return WinDefWindowProc(hwnd, msg, mp1, mp2);
case WM_TIMER:
    /* Upon getting a timer message, the invalidate rectangle call */
    /* will cause a WM_PAINT message to be sent, enabling animation */
    WinInvalidateRect(hwnd, NULLHANDLE, NULL);
    return WinDefWindowProc(hwnd, msg, mp1, mp2);
case WM_PAINT:
    /* This is what is done for every frame of the animation */
    t += 1.0;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix();
    glRotatef(t, 1.0, 1.0, 1.0);
    glBegin(GL_TRIANGLE_FAN);
        glColor3fv(colors[0]);
        glVertex3fv(verts[0]);
        glColor3fv(colors[1]);
        glVertex3fv(verts[1]);
        glColor3fv(colors[2]);
        glVertex3fv(verts[2]);
        glColor3fv(colors[3]);
        glVertex3fv(verts[3]);
        glColor3fv(colors[4]);
        glVertex3fv(verts[4]);
        glColor3fv(colors[1]);
        glVertex3fv(verts[1]);
    glEnd();
    glBegin(GL_TRIANGLE_FAN);
        glColor3fv(colors[5]);
        glVertex3fv(verts[5]);
        glColor3fv(colors[1]);
        glVertex3fv(verts[1]);
        glColor3fv(colors[4]);
        glVertex3fv(verts[4]);
        glColor3fv(colors[3]);
        glVertex3fv(verts[3]);
        glColor3fv(colors[2]);
        glVertex3fv(verts[2]);
        glColor3fv(colors[1]);
        glVertex3fv(verts[1]);
    glEnd();
    glPopMatrix();
    pglSwapBuffers(hab, hwnd);
    return WinDefWindowProc(hwnd, msg, mp1, mp2);
case WM_CHAR:
    mycode = (USHORT)SHORT1FROMMP(mp1);
    if ((mycode & KC_CHAR) && !(mycode & KC_KEYUP))
```

Figure 4. A simple OpenGL program using PGL interface (continued on page 44).



```
key = CHAR1FROMMP(mp2);
else if ((mycode & KC_VIRTUALKEY) && !(mycode & KC_KEYUP))
key = CHAR3FROMMP(mp2);
if (key == PM_ESCAPE)
WinPostMsg(hwnd, WM_CLOSE, (MPARAM)0, (MPARAM)0);
return WinDefWindowProc(hwnd, msg, mp1, mp2);
default:
return WinDefWindowProc(hwnd, msg, mp1, mp2);
}
}
main(int argc, char **argv)
{
PVISUALCONFIG vishead; /* visual configuration */
HMQ hmq; /* message queue */
HWND hwnd;
HWND hwndFrame;
ULONG createflags = FCF_TITLEBAR |
FCF_SYSMENU |
FCF_MINMAX |
FCF_SIZEBORDER;
QMSG qmsg; /* message */
HGC hgc; /* OpenGL context */
int major, minor; /* OpenGL version */
int err;

hab = WinInitialize(0);

/* Check to see if OpenGL exists */
if (pglQueryCapability(hab)) {
pglQueryVersion(hab, &major, &minor);
/* Version 1.0 */
if ((major == 1) && (minor == 0)) {
/* Choose a visual configuration that matches desired */
/* attributes in attriblist */
vishead = pglChooseConfig(hab, attriblist);
if (!vishead)
DispError("Couldn't find a visual!\n");
hmq = WinCreateMsgQueue(hab, 0);
if (!hmq)
DispError("Couldn't create a message queue!\n");
if (WinRegisterClass(
hab,
(PSZ)"PGLtest",
WindowProc,
CS_SIZEREDRAW | CS_MOVENOTIFY, /* Need at least this! */
0))
{
hwndFrame = WinCreateStdWindow (
HWND_DESKTOP, /* Child of the desktop */
WS_VISIBLE, /* Frame style */
&createflags, /* min FCF_MENU|FCF_MINMAX */
(PSZ)"PGLtest", /* class name */
"OpenGL Sample", /* window title */
WS_VISIBLE, /* client style */
0, /* resource handle */
1, /* Resource ID */
&hwnd); /* Window handle */
}
```

Figure 4. A simple OpenGL program using PGL interface (continued on page 45).



```
if (!hwndFrame)
    DispError("Couldn't create a window!\n");
/* you must set window size before you call pglMakeCurrent */
if (!WinSetWindowPos(
    hwndFrame,
    HWND_TOP,
    0,
    0,
    300,
    300,
    SWP_ACTIVATE | SWP_SIZE | SWP_MOVE | SWP_SHOW))
    DispError("Couldn't position window!\n");
hgc = pglCreateContext(hab, /* anchor block handle */
    vishead, /* visual configuration */
    (HGC)NULL, /* (no) shared contexts */
    (BOOL)TRUE); /* direct (fast) context */
if (!hgc)
    DispError("Couldn't create an OpenGL context!\n");
if (!pglMakeCurrent(hab, hgc, hwnd))
    DispError("Could not bind OpenGL context to window!\n");
/* Don't subclass your window past here! */
Setup();
/* Start timer to cause WM_TIMER messages to be sent */
/* periodically. This is used to animate. */
WinStartTimer(hab, hwnd, 0L, 0L);
while (WinGetMsg(hab, &qmsg, NULLHANDLE, 0, 0))
    WinDispatchMsg(hab, &qmsg);
}
}
}
}
```

Figure 4. A simple OpenGL program using PGL interface (continued from page 44).



*The relational and object paradigms appear to be in conflict, yet synergy exists between them. This article explores the integration of these technologies in a sample application. **By LISTON TATUM***

Engineering Object Persistence with C-Set++ and DB2/2



Liston Tatum

The purpose of this article is to show how readily DB2 lends itself to the persistent storage of object instance data and, conversely, how C++ can be used to enhance the expressive power of the relational algebra. In an effort to make this technology more accessible, and perhaps dispel a few myths, our discussion will center around a small prototype application. We'll begin with a brief description of the user interface classes, then move on to address some of the issues involved with the integration of DB2 into a C++ application. Along the way, we will consider the combination in the context of standards proposed by the Object Database Management Group, which we defer to as the most precise definition of an object database management system currently available. This article assumes at least a modest familiarity with the relational and object paradigms.

TMap (Table Map) is a utility that displays some of the information in the DB2 system catalog tables. As you can see from the screen image in Figure 1, TMap's user interface looks like the familiar OS/2 Drives utility. It begins by displaying a tree view of table creators; selecting a creator node expands the tree, showing table names. Opening a

table name creates a details view listing the columns that make up the table. The interface is built from two graphical user interface (GUI) classes, `Tables` and `Details`, which are descendants of `IFrameWindow` from IBM's User Interface Class Library. Each creates an instance of an `IContainerControl`, which it fills with objects subclassed from `IContainerObject`. The single instance of the `Tables` class is the focal point of the application, controlling the creation of `Details` objects on an as requested basis. A more detailed description of the implementation of these classes, which is not germane to this discussion, can be found in the source code distribution.

The user interface classes get their information from the `Database` class, whose class interface declaration is reproduced as Figure 2. The creator and name of each relation (among other things) is maintained by DB2 in the system catalog table `sysstables`, created by `sysibm`, while the attributes of the various columns are kept in `sysibm.syscolumns`. The `Database` class, and the classes it defines, implement methods that invoke the C language routines that provide the interface between the C++ application and the DB2 database system. The real work of the `Database` class is done by access routines in C (a precompiler limita-

Oddly enough, the most productive application development tool for Windows isn't from Microsoft.®



Introducing VisualAge™ for Windows™ and IBM Smalltalk.

Now you have the power to develop industrial-strength client/server applications for the Windows environment in the blink of an eye.

VisualAge and IBM Smalltalk Version 2.0 extend IBM's powerful new vision of programming. They provide multiple platform support including Windows and OS/2,® connectivity to business-critical data and

applications, scalability to create applications from the desktop to the enterprise, a fully object-oriented development environment, and a robust team version for greater programming productivity.

Break the code barrier with VisualAge.

With the simplicity of visual construction, you can create complex applications with amazing speed. What's more, you get the added flexibility of a completely integrated Smalltalk object-oriented base.

The language of objects™

IBM Smalltalk, included with VisualAge, is now available separately. Smalltalk programmers now have the standards-compliant, integrated development environment they need to easily develop fully portable applications.

To order or for more information, call **1 800 IBM-CALL, Dept. SA005**. In Canada, call **1 800 565-SW4U, ext. 279**, or contact your favorite reseller.

SOFTWARE FOR
OBJECT-ORIENTED TECHNOLOGY



tion); we incorporate them into object semantics by defining class methods to invoke them.

The Object Database Management Group (ODMG) has proposed a set of standards that define the characteristics of, and the operations that can be performed on, an object database management system (ODBMS). Among these is the idea of an a priori instance of the `Database` class, although it must be opened before it can be used. TMap begins by creating an instance of the `Database` class, which simply establishes connection to DB2. The ODMG has also proposed C++ language bindings. In an express effort to make them consistent with existing conventions, they allow an implicit open when a database session is started. We'll therefore assign responsibility

for the DB2 CONNECT operation to our `Database` class constructor, while the destructor takes care of the RESET. ODMG standards, and the design of DB2/2, allow one open database per process. This fits nicely with the design goal of DB2: locational transparency (at which point there will be only one database); so we instantiate a named global instance of the class in the main routine and call it DB2.

The database entities of interest to us are described by the access classes `Relations`, `Attributes`, and `TabRem`. These are transaction classes. In the ODMG model, they would inherit from the `Transaction` class, which would provide them with start, commit, abort, and checkpoint operations. Again we make use of C++ object initializa-

tion. When we work with sets in DB2 (tables are sets of structures), we need to declare and open a cursor before we access the data.

Consider the EXEC SQL DECLARE CURSOR statement in Figure 3. The placement of the cursor declaration is arbitrary (it has file scope) as long as it is located before its first use. Opening the cursor is equivalent to the ODMG `Transaction` class's start method: concurrency locks are acquired, the SQL statement is executed, and the results table gets built. Closing the cursor performs a commit, releasing the locks and freeing the cursor for reuse. These operations map neatly into class constructors and destructors. In Figure 2, we declare a constructor for each of our database access classes that calls the respective C language open cursor

SOM

The Object Factory—IDL (Version 2.0)

The first OS/2 SOM tool continues to expand SOM development with the introduction of version 2.0. This feature-packed upgrade forges new ground in state-of-the-art OOI interfaces, allowing the fastest possible SOM development. The Object Factory Version 2.0 is \$250.

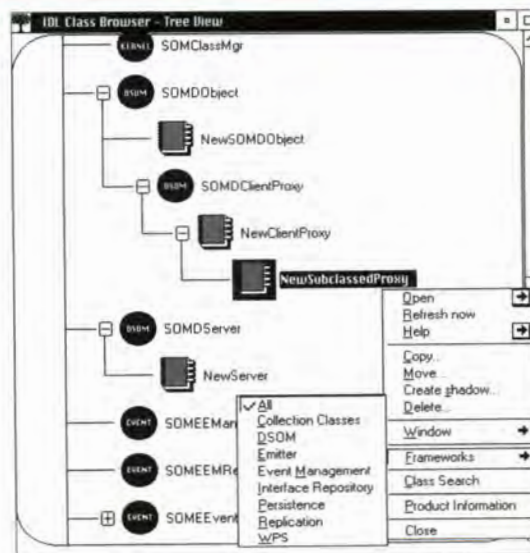
- Full-functioning SOM class browser
- Framework filters & Global class searching
- Complete IDL support
- Supports multiple inheritance
- Includes all SOM 2.0 classes and methods
- User-customizable class development environment
- Drag-drop backups & restorations

Also available from Synaptec:

The SOM Class Administrator — a complete runtime SOM administration utility presented in a hierarchical tree browser with the following features: Registers & Deregisters Classes, Creates & Deletes Instances, Optimizes System Performance and Handles DLL Version Control & Upgrades. The SOM Class Administrator is \$79.99. You never knew what was inside your machine! Can be used stand alone, or as an integrated component of The Object Factory.

Call (404)942-8699 for ordering information.

5085 Forest View Trail Douglasville Georgia 30135 (404)942-8699



SYNAPTEC, INC.

routine and a destructor to close them.

Applications that selectively update data should be provided with methods that execute explicit commits, known as checkpoints in the ODMG model. Frequent commits during a long-running selective update operation facilitate concurrent access by freeing locks. Also, they minimize the cost of abnormal program termination since update activity is rolled back to the last commit point. In this case, however, our database access transactions are read only. They go through the tables and end promptly, obviating the need for checkpoint processing. We mention it here to point out that the capability exists and can be implemented where appropriate.

Because a DB2 SELECT always

returns a (possibly empty) collection, the **Relations** and **Attributes** classes can be regarded as members of the ODMG collection type. In this case, both describe sets—or more precisely lists, since they have been sequenced by the **ORDER BY** predicates. **Relations** returns a list of userid-table name pairs, which it gets from `sysibm.sysables`; while **Attributes** defines the set of column names and attributes belonging to a specific table, which is selectively obtained from `sysibm.syscolumns`. In Figure 3, for example, a specific list of attributes is defined by the **SELECT** statement in the C function `openColRow()`, which is invoked by the **Attributes** constructor.

Collections require iterators to facilitate the transition between set-based operations and procedural item-at-a-time processing. Cur-

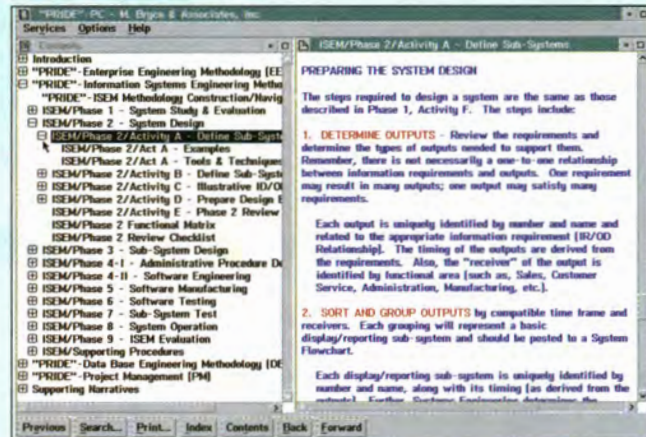
rently, the ODMG's object query language bindings support dynamic queries only. Their model specifies a `create_iterator()` method as a member of the virtual **Collection** class. While this is useful for ad hoc inquiries, our requirements are known in advance, allowing us to declare iterators as members of the collection classes. We'll call them `next()`, in deference to the ODMG's naming conventions. The methods call C functions that fetch the next row; they return the row not found condition code when we try to read past the end of the table.

There is a special case known as a singleton select, in which a DB2 select statement is guaranteed to return at most one row. In our example, **TabRem** defines a single valued set because it fetches the remark (if any) that corresponds to a



SAVE A TREE! BUY PRIDE®-PC

The world's most comprehensive development methodologies are now available on the OS/2 desktop. And at an affordable price, no developer or IS manager should be without it.



New from MBA, creators of structured methodologies, comes "PRIDE"-PC, proven methodologies for: **Enterprise Engineering, Information Systems Engineering** (integrating BPR with Software Engineering), **Data Base Engineering** (logical and physical), and **Project Management**. Using the standard OS/2 "View" utility, "PRIDE"-PC includes: Tutorials and instructions with hypertext, sample deliverables, functional matrices, checklists, and more. And at a price of just \$150 per PC, *why pay more?* For information and/or a demo disk, call:

M. BRYCE & ASSOCIATES, INC.

777 Alderman Road, Palm Harbor, FL 34683 USA

Tel: 813/786-4567 • Fax: 813/786-4765 • BBS: 813/786-4864 • Internet: TimB1557@aol.com

Circle Reader Service Number 22

specific creator.name pair (which must be unique within a database). Note the `SELECT INTO` statement in Figure 3, which loads data directly into host language variables, eliminating the need for a cursor. This is particularly useful with the aggregate functions (min, max, avg, count, and sum). `TabRem` complies with ODMG standards in that it defines a (single valued) set and is a transaction class. We'll use the implicit constructor and destructor in this case, while the iterator is specialized for matching keyed retrieval: `getTabRem()` returns the remark that corresponds to the creator and name key fields.

Let's examine some of the mechanical details of the application programming interface to DB2. The interface requires that a buffer

be identified to handle the data involved in a transaction. Consider the `EXEC SQL BEGIN DECLARE SECTION - END DECLARE SECTION` statement pairs in Figure 3. This is where we tell the `SQLPREP` precompiler the names of the pointers we'll be using to refer to our data areas. We've also allocated some storage, although we may not use it. The precompiler (`SQLPREP`) uses this allocation to determine the length of the data we expect to transfer. Remember, these declarations have file scope; including them in the body of the function is a notational convenience.

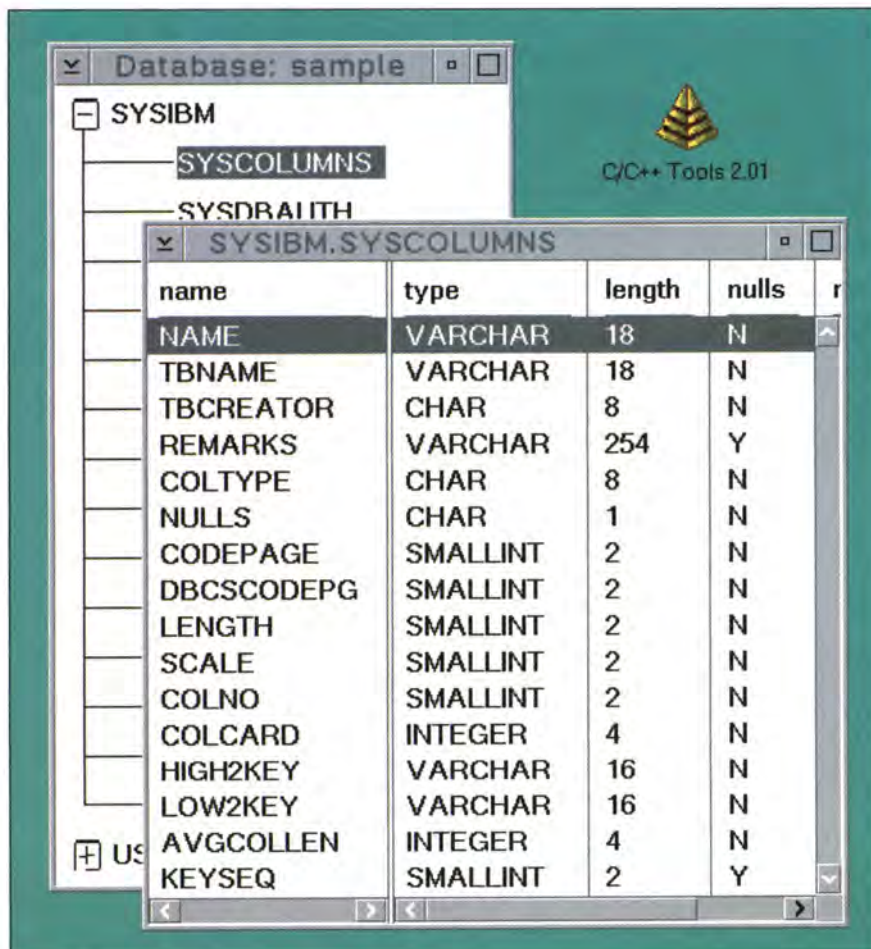
`SQLPREP` reads the `.sqc` file, converting the `EXEC SQL` statements to comments and replacing them with calls to the DB2 function library, while writing the expanded file to

disk with a `.c` extension. In addition, it creates a database request module (DBRM) that references our pointer variables. Our database access methods pass the address of a data structure allocated in the caller's frame of reference to the C language functions. The C code sets the pointers known to the DBRM to point into the caller's address space (so we avoid having to copy the data) then executes the `SQL FETCH` or `SELECT INTO`. Aside from the signatures of the functions involved, these data structure definitions are the only application code held in common by the C and C++ units of translation.

If an attribute allows nulls (that is, its value may be undefined for a given instance), an implicit null indicator corresponding to the column must be checked before using the data. Consider the `EXEC SQL SELECT INTO` statement in `selectTabRem()`. Note the use of the null indicator, `tbrem_nid`, immediately following (without a comma separator) the `ptbrem` remarks field pointer. If the value of the null indicator is less than zero, the data in the field is left over from the last time it had a defined value and should not be used. Here we truncate the character string array.

The results of DB2 operations are reflected in the SQL communications area (`sqlca`) structure. The `sqlca.sqlcode` value must be checked after each operation. For simplicity, we define an `EMsg` class (not shown) to handle exceptions by looking up the corresponding message text, then terminating the application. Some operations should be retried (for example, enqueue lockouts) in a production system. You may wish to handle other exceptions more gracefully.

The most important difference between this configuration and object database management systems is that although we can store instances of the fundamental data



The screenshot shows the TMap user interface. On the left, a tree view displays the database structure: 'Database: sample' contains 'SYSIBM', which contains 'SYSCOLUMNS' and 'SYSDBAITH'. The 'SYSCOLUMNS' table is selected, and its details are shown in a table view on the right. The table view has columns for 'name', 'type', 'length', and 'nulls'. The data rows are as follows:

name	type	length	nulls
NAME	VARCHAR	18	N
TBNAME	VARCHAR	18	N
TBCREATOR	CHAR	8	N
REMARKS	VARCHAR	254	Y
COLTYPE	CHAR	8	N
NULLS	CHAR	1	N
CODEPAGE	SMALLINT	2	N
DBCSCODEPG	SMALLINT	2	N
LENGTH	SMALLINT	2	N
SCALE	SMALLINT	2	N
COLNO	SMALLINT	2	N
COLCARD	INTEGER	4	N
HIGH2KEY	VARCHAR	16	N
LOW2KEY	VARCHAR	16	N
AVGCOLLEN	INTEGER	4	N
KEYSEQ	SMALLINT	2	Y

Figure 1. TMap user interface.



types, there is no direct way to maintain the definition of a user-defined type in the database—at this time. Consider a patient's age, which we choose to define as the difference between today's date and their date of birth. The definition of the `patientAge` type is procedural, as are operations like `patientAge + 1`. The extensions the Toronto lab is incorporating in DB2/2 Version 2 will mitigate this problem. User-defined types and operations can't be supported directly by the database without them.

An ODMG-compliant ODBMS provides persistent storage for the bag, set, list, and array collection classes. Of these, only the set is supported by the SQL data definition language. A bag (a set with repeating elements) can be created, but its use is strongly discouraged because it doesn't participate correctly in set-based operations. Although a table might look like a list, there is no presumed order to the elements of a relation, and certainly no support for obtaining the *n*th row, as in an array. Interestingly, collection types can be defined in SQL's data manipulation language. The results of a `SELECT` may include duplicates; `SELECT creator FROM sysibm.systables` describes a bag. `SELECT DISTINCT` returns a (possibly null) set. The predicate `ORDER BY` defines a list ordered by the contents of the data. The issue is one of early vs. late bindings—whether to store the data as a list, for example, or sort it on retrieval.

Traversal paths cannot be explicitly created in a relational system, although relationships that must be enforced between tables are defined as referential integrity (RI) constraints. In our example, nothing in `sysibm.systables` suggests where to look for information about the set of attributes belonging to each of the relations named

therein. The `Attributes` class defines this set, which is the traversal from `Relations` to `Attributes`. In this case, the RI constraints that guarantee `Attributes` will never return an empty set comes with the DB2 implementation; we can do the same for relations we create. DB2 tracks these transitive relationships in a separate table (`sysibm.sysrels`), however, which makes their use for navigational purposes somewhat inconvenient.

In contrast, an ODBMS provides a way to express the application semantics explicitly in the schema of the database (a table has columns, for example). The ODMG's draft standard proposes an addition to C++ syntax that would allow the expression of set-based relationships between classes. We have demonstrated we can do the same here. The differ-

ence, of course, is that our classes are not part of the database, although the RI constraints upon which they depend are. The practical consequence is that we need to create a class library to supply the persistent objects of our application domain while restricting bind authority. This ensures that modifications of instance data are accomplished only through methods defined to the class.

A synergistic relationship exists between these paradigms that roughly parallels the need for the friend construct in C++. We can use them in concert to describe the persistent objects of our application domain, without precluding future aggregations. This is important because there is no definitive way to categorize an object. A useful grouping in one context will be awkward in another.

```
class Database {
public:
    Database(char *db) { if (connect(db)) EMsg(); }
    ~Database() { reset(); }
    class Relations { //creator, tablename from sysibm.systables
    public:
        Relations() { if(openTabRow()) EMsg(); }
        int next() { return fetchTabRow(&tRow); }
        ~Relations() { closeTabRow(); }
        struct tabRow tRow;
    };
    class Attributes { //column details from sysibm.syscolumns
    public:
        Attributes(char *creator, char *name)
            { if(openColRow(creator, name)) EMsg(); }
        int next() { return fetchColRow(&cRow); }
        ~Attributes() { closeColRow(); }
        struct colRow cRow;
    };
    class TabRem { //remark from sysibm.systables
    public:
        void getTabRem(char *creator, char *name, char(*tr)[255])
            { if (selectTabRem(creator, name, tr)) EMsg(); }
    };
};
```

Figure 2. Database class interface declaration from `TMap.hpp`.



While the rules of normalization dictate a context-free data structure, the object paradigm provides for abstraction, including the expression of transitional (is a, has a, etc.) relationships. These relationships provide the meaning of the data in context and make it possible to express the interdependencies that describe a particular system.

We began with a brief description of an application based on the container control from the IBM User Interface Class Library. We then discussed the interrelationship between DB2/2 and C-Set++, touching on both mechanical details and theoretical considerations, all of which we examined in the context of emerging standards for object database management systems. We hope we have demonstrated some of the basics of using C-Set++ and DB2/2 to implement persistent objects.

TMap was compiled using version 2.01 of IBM's C-Set++ Compiler and User Interface Class Library. DB2/2 is also required, as is OS/2 2.1, due to a bug in OS/2 2.0's container tree view. The source is available in machine-readable form from the OS/2 Developer forum on CompuServe, the OS2BBS on IBMLink, or directly from the author via e-mail.

Liston Tatum, *University of Virginia Health Sciences Center, Charlottesville, VA 22098*, began his career as a systems engineer in minicomputer and mainframe operating systems. He then became involved with telecommunications and, more recently, database management systems. This eclectic background provides the context for his professional interest in C++ and the relational and object paradigms. He is currently a senior systems engineer and database administrator with the Information Systems Division of the University's Health Sciences Center. He can be reached on Internet at glt@dmt03.mcc.Virginia.EDU, or via IBMMail as USUVASHB.

```
int openColRow(char (*cre)[9], char (*nam)[19]) { //sysibm.syscolumns
    ptbcreator = cre; ptbname = nam;
    EXEC SQL DECLARE ColRow CURSOR FOR
        SELECT NAME, REMARKS, COLTYPE, LENGTH, NULLS, COLNO
        FROM SYSIBM.SYSCOLUMNS
        WHERE TBCREATOR = :ptbcreator AND TBNAME = :ptbname
        ORDER BY COLNO;
    EXEC SQL OPEN ColRow;
    return sqlca.sqlcode;
}

int fetchColRow(struct colRow *c) { //get row from sysibm.syscolumns
    EXEC SQL BEGIN DECLARE SECTION;
    char (*pcname)[19];           /* name varchar(18) */
    char (*pcrem)[255];           /* remarks varchar(254) */
    short crem_nid;               /* remarks null indicator */
    char (*pctype)[9];            /* coltype char(8) */
    short clength;                /* length smallint */
    char (*pcnulls)[2];           /* nulls char(1) */
    EXEC SQL END DECLARE SECTION;
    pcname = &c->name; pcrem = &c->rem; //point at fields in
    pctype = &c->type; pcnulls = &c->nulls; //syscolumn row structure
    EXEC SQL FETCH ColRow INTO
        :pcname, :pcrem :crem_nid, :pctype, :clength, :pcnulls;
    c->length = clength;
    if (crem_nid < 0) c->rem[0] = NULL; //null remark? truncate!
    return sqlca.sqlcode;
}

void closeColRow(void) { EXEC SQL CLOSE ColRow; } //sysibm.syscolumns

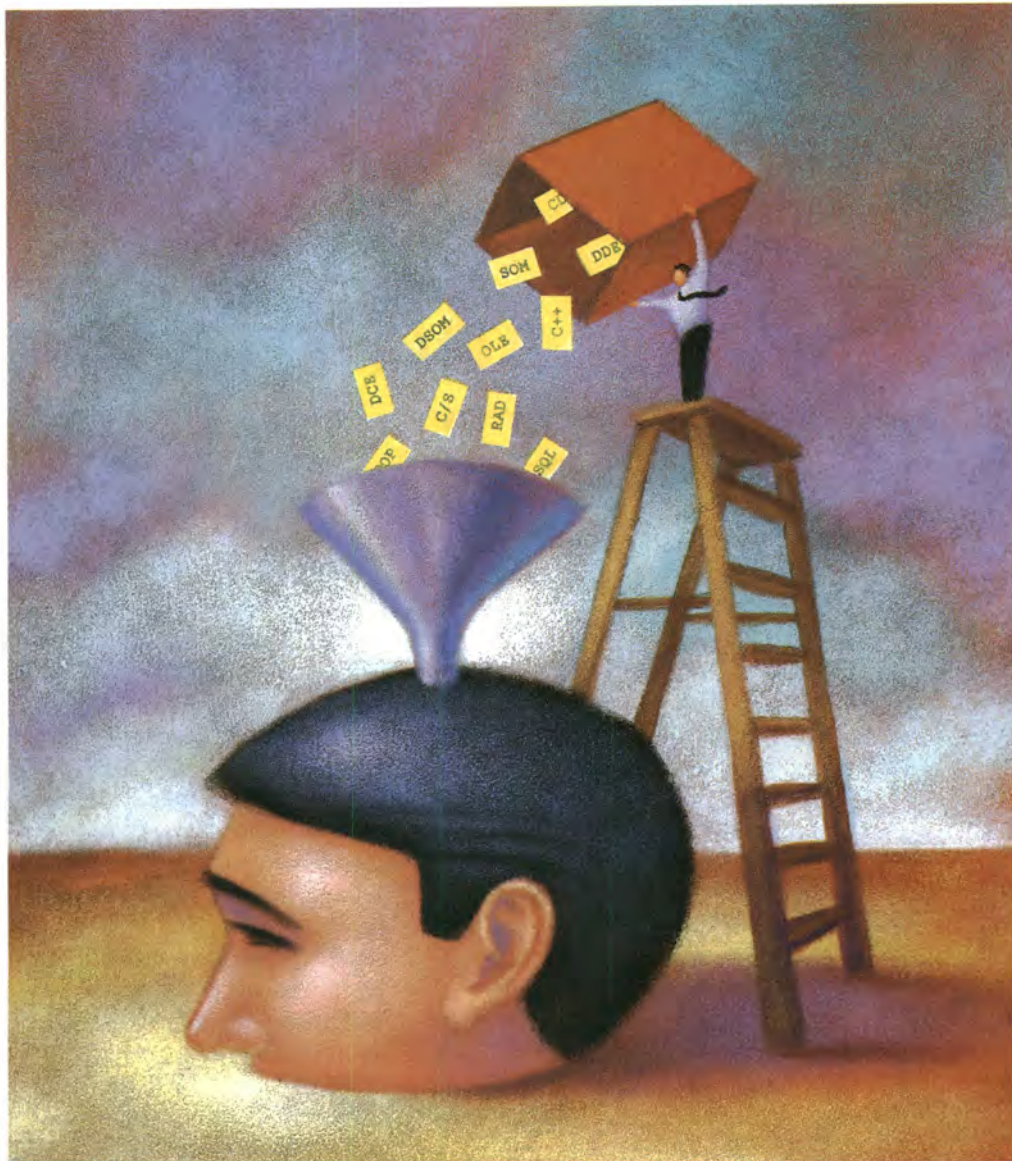
int selectTabRem //get remark on creator.name from sysibm.systables
(char (*creator)[9], char (*name)[19], char (*rem)[255]) {
    EXEC SQL BEGIN DECLARE SECTION;
    char (*ptbrem)[255];          /* remarks on table, if any */
    short tbrem_nid;              /* null indicator */
    EXEC SQL END DECLARE SECTION;
    ptbrem = rem; ptbcreator = creator; ptbname = name;
    EXEC SQL SELECT REMARKS INTO :ptbrem :tbrem_nid
    FROM SYSIBM.SYSTABLES
    WHERE CREATOR = :ptbcreator AND NAME = :ptbname;
    if (tbrem_nid < 0) *rem[0] = NULL; //truncate?
    return sqlca.sqlcode;
}
```

Figure 3. Sample Database class method implementations from `db.sqc`.

REFERENCES

- Loomis, Mary E. S., et al. "The ODMG Object Model", *Journal of Object Oriented Programming* 6(3):64-69 (June 1992), SIGS Publications, New York, NY
- Strickland, Henry. "ODMG-93, The Object Database Standard for C++", *C++ Report* 5(8):45-48,70 (October 1993), SIGS Publications, New York, NY
- Cattell, R. G. G., ed. *The Object Database Standard ODMG-93*, Morgan Kaufmann, San Mateo, CA 1994

Attend the only show that has all the tools you need to make your development vision a reality.



Your mind is your livelihood. And in a world of converging technologies, you need to fill it with more than just the buzzwords. New technologies are supposed to solve your development problems. Yet for every problem solved, *you* know a new one will emerge.

Software Development '95 takes you beyond the buzzwords—from component to full-scale enterprise application development, we show you how to create tangible, real-world solutions.

We explore the issues surrounding object-oriented

programming, client/server migration, rapid application development, and more. You get a clear picture of where the technology is—and where it's headed. SD '95 delivers all the information, tools, and business contacts you need to thrive in today's challenging development world.

Don't miss out. Bring your vision to SD '95. And leave with the tools to make it a reality.

For more information, call (800) 441-8826, or (415) 688-4345 for fax-back info, or e-mail us at sd95west@mfi.com.



Software Development '95, February 13–17, Moscone Convention Center, San Francisco, CA

250 CLASSES + 250 VENDORS + 15,000 DEVELOPMENT PROFESSIONALS

OS2D11



This article discusses how to implement a custom line style package and integrate it with the OS/2 Presentation Manager GPI. By NICK HODAPP

Line Styles and the OS/2 PM GPI

Several months ago, my company decided to develop viewing software to provide access to graphics stored in a proprietary vector graphic data file. The graphics we wished to display would have been created on a computer-aided design (CAD) system, which has roots in the DOS world and, more recently, in the 16-bit OS/2 arena. Our goal was to provide our customers with an elegant solution to access their data in Presentation Manager rather than through an out-dated DOS full-screen environment.

About a month into our development, we hit our first major snag. The file format we were reading supports user-definable line styles. Pages 3-4 of the *OS/2 2.0 PM Graphics Programming Guide* states, "You cannot define other line types for PM applications." Upon showing this to my supervisor, he said we had to either support complex line types or halt development. I nodded solemnly and put the problem on the back burner for several weeks.

There is a rationale behind the problem. Software written for DOS and other older platforms generally had direct access to the display hardware itself, mainly Enhanced Graphics Adapter (EGA) or Video Graphics Array (VGA). Routines could be easily coded in as-

sembly language to plot lines of any style or pattern. These systems were fast and reliable, until the user bought an SVGA card. It was the software developer's responsibility to provide support for multiple display configurations. With the advent of OS/2 Presentation Manager (and other systems as well), developers have come to rely on the operating system to provide a standard way of talking to the video hardware, whether it be VGA, XGA, MGA, or any other "standard." Within Presentation Manager, this communication is through the graphical programming interface (GPI). Operating systems, in turn, rely on the hardware vendors to provide a driver that supports the GPI command set for their particular video cards. Unfortunately, the GPI does not provide a way to customize how a line is drawn (the line type or style). Instead, the GPI provides nine standard line types to which all graphics packages must conform.

A line style is a pattern that is repeatedly drawn along the length of the displayed line. Therefore, a solution to this problem is to manually plot the individual pattern along the entire path of the desired line. The trick is to implement this efficiently and seamlessly into the Presentation Manager environment. "Efficiently" is more difficult than "seamlessly."



To implement a custom line style package, methods are required for defining individual line style patterns, selecting line styles for use, and plotting lines with custom styles. Once in place, the functionality can be integrated with the GPI. In the following paragraphs, the terms line style pattern and line style refer to the actual pattern that is repeated along the length of the output line segment.

The line style pattern data will be stored as an array of disjoint line segment points. When plotted, these points are transformed to new positions along the path of the output line and then connected with a single call to `GpiPolyLineDisjoint()`. Plotting with `GpiPolyLineDisjoint()` is the most efficient method of drawing several line segments to a presentation space. However, I have found a few display drivers that do not properly interpret calls to `GpiPolyLineDisjoint()`. Until such drivers are fixed, the calling software must revert to making use of `GpiMove()` and `GpiLine()`. The structures in Figure 1 are used to define a custom line style. Note that unlike the predefined GPI line styles, this implementation supports two-dimensional lines.

For ease of definition, I have written the `GpiLoadCustomStyle()` function to load a line style defined in the application resource fork. `GpiLoadCustomStyle()` stores the style data in an array of styles and returns the array offset (+10) for reference. Figure 2 defines the RCDATA format used to describe custom styles. `GpiFreeCustomStyle()` frees and deletes a custom line style.

To keep track of the currently selected line style, calls to `GpiSetLineStyle()` are intercepted by `nphSetLineStyle()`. This replacement function calls `GpiSetLineStyle()` for line style types 9 or less, and it calls `GpiSetLineStyle()` with the `LINE_TYPE_SOLID` attribute for custom line styles (10 and above). A global variable tracks

the actual current line style. This implementation fails to track separate line style settings for multiple presentation spaces; however, such functionality is feasible to implement.

The `nphLine()` function is a replacement function for `GpiLine()`. When called, `nphLine()` checks the current line style, calls `GpiLine()`, and returns if the current style is predefined. If the current line style is custom, `nphLine()` proceeds to construct a disjoint array of line segments to make up the custom line. When this process is complete, `GpiPolyLineDisjoint()` plots the line to the desired presentation space.

```
typedef struct lineSegStruct {
    POINTL    point1,
             point2;
} LINESEG, *PLINESEG;

typedef struct styleStruct {
    USHORT    usSegWidth; /* Pattern width      */
                      /* in units          */
    USHORT    usNumSegs; /* Number of segments */
                      /* in pattern         */
    PLINESEG  segments; /* The segment data   */
} STYLE, *PSTYLE;

A zig-zag line style could be initialized with the following:

STYLE zigzag;
zigzag.usSegWidth = 40;
zigzag.usNumSegs = 3;
zigzag.segments = (PLINESEG)malloc(3 * sizeof(LINESEG));
zigzag.segments[0].point1.x = 0; zigzag.segments[0].point1.y = 0;
zigzag.segments[0].point2.x = 10; zigzag.segments[0].point2.y = 10;
zigzag.segments[1].point1.x = 10; zigzag.segments[1].point1.y = 10;
zigzag.segments[1].point2.x = 30; zigzag.segments[1].point2.y = -10;
zigzag.segments[2].point1.x = 30; zigzag.segments[2].point1.y = -10;
zigzag.segments[2].point2.x = 40; zigzag.segments[2].point2.y = 0;
```

Figure 1. Custom line style implementation.



It is necessary to first calculate the length of the output line segment. This length determines the number of times the line style pattern is to be drawn along the output path.

Next, the sine and cosine of the output line segment angle are determined. These values are used to transform the line style data points to correct locations along the output path.

Once the required values are calculated, `nphLine()` constructs the output line. An outer for loop counts the number of times the pattern is drawn along the line segment path. For each iteration, an inner for loop transforms each line style point to the correct location along the output line path. Upon completion, a straight line "tail" is attached to make the line segment output to the required length. Finally, the completed line is output with a call to `GpiPolyLineDisjoint()`.

For the functions `nphSetLineType()` and `nphLine()` to intercept the corresponding GPI calls, a header file redefines `GpiSetLineType()` and `GpiLine()` to call the replacement functions. The demonstration source includes a window function that demonstrates the use and output of the discussed functionality.

Full integration with the GPI requires additional code to support `GpiPolyLine()`, `GpiPolyLineDisjoint()`, `GpiPointArc()`, `GpiBox()`, and so on. It is also necessary in most situations to track separate line style settings for each presentation space used; this requires intercepting more GPI calls. Hopefully, future updates to the GPI will add support for some type of line style definition. In the meantime, implementations such as this offer acceptable output with little performance degradation.

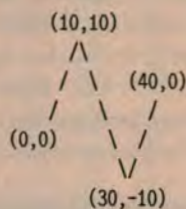
Nick Hodapp is a senior programmer with Power System Engineering Inc. in

Line style data may be defined in the application resource fork by means of RCDATA. The format for this data follows:

```
RCDATA RESOURCE_ID
{
  SEGS_IN_PATTERN, PATTERN_WIDTH, x1, y1, x2, y2, x3, y3, x4, y4, ...etc.
}
```

Where `RESOURCE_ID` is the assigned resource id, `SEGS_IN_PATTERN` is the number of line segments that make up the line style pattern, and `PATTERN_WIDTH` is the maximum width of the pattern in units. The coordinates for the style pattern follow. The zig-zag pattern could be described with:

```
RCDATA ZIGZAG
{
  3, 40, 0, 0, 10, 10, 30, -10, 40, 0
}
```



Note that the definition centers the y axis on coordinate 0, the x axis on `PATTERN_WIDTH / 2`. If the defined `PATTERN_WIDTH` is less than the actual span of the pattern, overlap will occur when the output line is plotted.

Figure 2. RCDATA format used to describe custom styles.

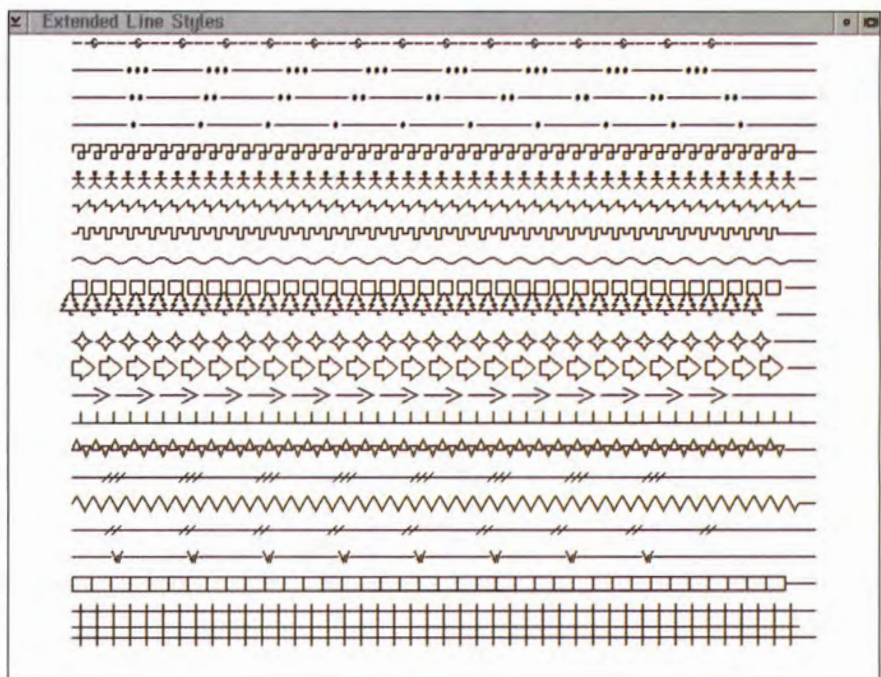


Figure 3. Line style output.

Madison, Wis. He is the architect of PSE's PSMAP software, which provides viewing and redlining of CableCad files for electric, telephone, and gas utilities worldwide. He presented two GPI-oriented classes at the 1994 ColoradOS/2 conference.

REFERENCES

OS/2 2.0 PM Graphics Programming Guide, ISBN: 1-56529-156-5.

Complete sample code is available on CompuServe's OS2DF2 Forum, OS/2 Developer section. Download file CLINES.ZIP.



```
CUSTSTL.H

BOOL GpiLoadLineStyle(HMODULE, LONG);
BOOL GpiFreeLineStyle(LONG);

BOOL nphSetLineType(HPS, LONG);
LONG nphLine(HPS, POINTL);

#define GpiSetLineType(a,b) nphSetLineType(a,b)
#define GpiLine(a,b) nphLine(a,b)
```

Figure 4. CUSTSTL.H.

```
CUSTSTL.CPP

#define INCL_DOS
#define INCL_GPI

#include <os2.h>
#include <math.h>
#include <stdlib.h>
#include <memory.h>

#define MAX_INTERNAL_LINE_SEGS 1000
#define MAX_STYLES 200
#define CUSTOM_STYLE_OFFSET 10

/* Round x up */
#define ROUND(x) x - (long)x > 0.5 ? x + 1 : x

/* Custom line style structures */
typedef struct lineSegStruct {
    POINTL point1,
           point2;
} LINESEG, *PLINESEG;

typedef struct styleStruct {
    USHORT usSegWidth;
    USHORT usNumSegs;
    PLINESEG segments;
} STYLE, *PSTYLE;

/* Global line style data */
STYLE styleList[MAX_STYLES];

LONG glCurrLineStyle;

POINTL lineOut[MAX_INTERNAL_LINE_SEGS];

/* Loads a line style from the supplied resource file */
```

Figure 5. CUSTSTL.CPP (continued on page 58).



```
/* Returns line assigned style number */
/* Resource format: */
/* RCDATA Resource_Id { */
/* NumSegs, StyleWidth, Seg1.x1, Seg1.y1, Seg1.x2, Seg1.y2, */
/* Seg2.x1, Seg2.y1, Seg2.x2, Seg2.y2, ... */
/* } */

LONG GpiLoadLineStyle(HMODULE resource, LONG resourceID)
{
    ULONG currStyle, i, j;
    PSHORT resourceData;

    static BOOL firstLoad = TRUE;

    /* Initialize styleList on first call only */
    if (firstLoad) {
        memset(styleList, 0, sizeof(styleList));
        firstLoad = FALSE;
    }

    /* Locate a free style offset */
    for (currStyle = 0 ; currStyle < MAX_STYLES ; currStyle++)
        if (!(styleList[currStyle].usNumSegs)) break;

    if (currStyle == MAX_STYLES) return 0L;

    /* Load the resource */
    DosGetResource(resource, RT_RCDATA, resourceID, (PPVOID)&resourceData);

    /* Set usNumSegs, allocate usNumSegs LINESEG's */
    styleList[currStyle].usNumSegs = resourceData[0];
    styleList[currStyle].segments = (PLINESEG)malloc(*resourceData * sizeof(LINESEG));

    /* Copy the style width */
    styleList[currStyle].usSegWidth = resourceData[1];

    /* Copy the style segment points */
    for (j = 0, i = 2 ; j < resourceData[0] ; j++) {
        styleList[currStyle].segments[j].point1.x = resourceData[i++];
        styleList[currStyle].segments[j].point1.y = resourceData[i++];
        styleList[currStyle].segments[j].point2.x = resourceData[i++];
        styleList[currStyle].segments[j].point2.y = resourceData[i++];
    }

    /* Free the resource */
    DosFreeResource(resourceData);

    /* Return the assigned style number */
    return currStyle + CUSTOM_STYLE_OFFSET;
}

/* Frees the data associated with a custom line style so that
/* the style number may be re-used. */
/* Returns FALSE on failure */
BOOL GpiFreeLineStyle(LONG lStyle)
{
    /* Predefined style returns FALSE */
    if (lStyle < CUSTOM_STYLE_OFFSET) return FALSE;

    /* Free the style data */
    lStyle -= CUSTOM_STYLE_OFFSET;
    styleList[lStyle].usNumSegs = 0;
    styleList[lStyle].usSegWidth = 0;
    free(styleList[lStyle].segments);

    /* return OK */
}
```

Figure 5. CUSTSTL.CPP (continued on page 59).



```
    return TRUE;
}

/* Replacement for GpiSetLineStyle() */
/* Sets line type. Does not perform management of different */
/* styles between separate PS's except for default predefined */
/* line styles... */
/* Returns FALSE on failure */
BOOL nphSetLineStyle(HPS hps, LONG lStyle)
{
    ULONG i;

    // Set global glCurrLineStyle variable
    glCurrLineStyle = lStyle;

    /* Determine if requested line style is GPI or custom */
    if (lStyle < CUSTOM_STYLE_OFFSET)
        return GpiSetLineStyle(hps, lStyle);

    /* Set GPI line style to solid */
    return GpiSetLineStyle(hps, LINETYPE_SOLID);
}

/* Replacement for GpiLine(). Checks current line style, then */
/* plots accordingly. Predefined line styles are routed to */
/* normal GpiLine(). */
/* Returns correlation/error code from GpiLine(), */
/* GpiPolyLineDisjoint() */
LONG nphLine(HPS hps, PPOINTL pEndPoint)
{
    POINTL startPoint;

    LONG    startX,
            startY,
            lStyle,
            opp, adj,
            currSegLength,
            j, k, l;

    double  sinSegAngle,
            cosSegAngle;

    /* Check to see if the current line style is GPI or custom */
    /* If GPI, draw the line segment normally */
    if (glCurrLineStyle < CUSTOM_STYLE_OFFSET)
        return GpiLine(hps, pEndPoint);
    else
        lStyle = glCurrLineStyle - CUSTOM_STYLE_OFFSET;

    /* Determine the current position within the ps */
    if (!(GpiQueryCurrentPosition(hps, &startPoint)))
        return FALSE;

    /* Determine opp/adj sides to virtual triangle */
    opp = pEndPoint->y - startPoint.y;
    adj = pEndPoint->x - startPoint.x;

    /* Calculate output line segment length in units */
    /* Use an integer sqrt() function here for speed... */
    currSegLength = (LONG)sqrt((double)((opp * opp) + (adj * adj)));
    if (!(currSegLength)) return 0L;

    /* Calculate sin/cos of the angle set by the output line */
    sinSegAngle = (double)(double)opp / (double)currSegLength;
    cosSegAngle = (double)(double)adj / (double)currSegLength;
```

Figure 5. CUSTSTL.CPP (continued on page 60).



```
/* Loop through line style sub-segments */
for (j = 0, l = 0 ; j < (currSegLength / styleList[lStyle].usSegWidth) - 1 ; j++) {

    /* Calculate current segment starting position */
    startX = (j * styleList[lStyle].usSegWidth * cosSegAngle) + startPoint.x;
    startY = (j * styleList[lStyle].usSegWidth * sinSegAngle) + startPoint.y;

    /* Loop through current sub-segment points to translate */
    /* and rotate to correct position along output path */
    for (k = 0 ; k < styleList[lStyle].usNumSegs ; k++) {

        lineOut[l].x = ROUND( ((styleList[lStyle].segments[k].point1.x * cosSegAngle - \
                               styleList[lStyle].segments[k].point1.y * sinSegAngle) + startX));
        lineOut[l++].y = ROUND( ((styleList[lStyle].segments[k].point1.x * sinSegAngle + \
                                   styleList[lStyle].segments[k].point1.y * cosSegAngle) + startY));

        lineOut[l].x = ROUND( ((styleList[lStyle].segments[k].point2.x * cosSegAngle - \
                               styleList[lStyle].segments[k].point2.y * sinSegAngle) + startX));
        lineOut[l++].y = ROUND( ((styleList[lStyle].segments[k].point2.x * sinSegAngle + \
                                   styleList[lStyle].segments[k].point2.y * cosSegAngle) + startY));
    }
}

/* Add a straight trailer to end of output line segment */
lineOut[l].x = (j * styleList[lStyle].usSegWidth * cosSegAngle) + startPoint.x;
lineOut[l++].y = (j * styleList[lStyle].usSegWidth * sinSegAngle) + startPoint.y;
lineOut[l].x = pEndPoint->x;
lineOut[l++].y = pEndPoint->y;

/* Plot the line with GpiPolyLineDisjoint() */
return GpiPolyLineDisjoint(hps, l, lineOut);

/* If driver incorrectly processes GpiPolyLineDisjoint(), */
/* revert to GpiMove() -> GpiLine()... */
/* for (j = 0 ; j < l ; j++) {
    GpiMove(hps, &lineOut[j]);
    GpiLine(hps, &lineOut[++j]);
}
*/}
```

Figure 5. CUSTSTL.CPP (continued from page 59).

```
RCDATA 106
{
    6, 12, 0, 0, 1, 3, 1, 3, 3, 5, 3, 5, 6, 6, 6, 6, 9, 5, 9, 5, 11, 3, 11, 3, 12, 0
}
RCDATA 108
{
    5, 60, 0, 0, 20, 0, 23, 0, 28, 0, 31, 0, 36, 0, 39, 0, 44, 0, 47, 0, 60, 0
}
RCDATA 123
{
    14, 16, 0, 0, -2, 0, -2, 0, -2, 3, -2, 3, -9, 3, -9, 3, -5, 7, -5, 7, -8, 7, -8, 7, -4, 11, -4, 11, -6, 11, -6, 11, -2, 16, -
    2, 16, 2, 11, 2, 11, 0, 11, 0, 11, 4, 7, 4, 7, 1, 7, 1, 7, 5, 3, 5, 3, -2, 3
}
```

Figure 6. Sample line style definitions (excerpt from full file).



```
#define INCL_WIN
#define INCL_GPI

#include <os2.h>
#include "linestl.h"
#include "custstl.h"

MRESULT EXPENTRY ClientWndProc (HWND hWnd, ULONG msg, MPARAM mp1, MPARAM mp2)
{
    ULONG i;
    HPS hps;
    POINTL line[2];
    RECTL rectl;

    static LONG maxStyle;

    switch (msg)
    {
    case WM_CREATE:
        for (i = STYLE_RES_START ; i <= STYLE_RES_END ; i++)
            maxStyle = GpiLoadLineStyle(NULLHANDLE, i);

        break;

    case WM_PAINT:
        hps = WinBeginPaint (hWnd, 0, 0);

        WinQueryWindowRect(hWnd, &rectl);
        WinFillRect(hps, &rectl, CLR_WHITE);

        line[0].x = 50;
        line[0].y = 15;
        line[1].x = rectl.xRight - rectl.xLeft - 50;
        line[1].y = 15;

        for (i = 20 ; i <= maxStyle ; i++) {
            GpiSetLineType(hps, i);
            GpiMove(hps, &line[0]);
            GpiLine(hps, &line[1]);

            line[0].y += 20;
            line[1].y += 20;
        }

        WinEndPaint (hps);
        break;

    case WM_SIZE:
        WinInvalidateRect(hWnd, NULL, TRUE);
        break;

    case WM_ERASEBACKGROUND:
        return (MRESULT) TRUE;

    case WM_DESTROY:
        for (i = 10 ; i <= maxStyle ; i++)
            GpiFreeLineStyle(i);

        break;
    }

    return WinDefWindowProc (hWnd, msg, mp1, mp2);
}
```

Figure 7. Sample window procedure, displays sample styles.

OS/2 Developer

Opt-Tech Sort/Merge

High Performance Sort/Merge/Select Utility

Use as a stand-alone utility or call as a Subroutine. Many features including unlimited file size, multiple keys, record selection, duplicate elimination, summing and more!

Call for free brochure Opt-TechData Processing
P.O. Box 678
Available for: Zephyr Cove, NV 89448
OS/2 or Unix \$249 Phone (702) 588-3737
DOS or Windows \$149 Fax (702) 588-7576

Circle Reader Service Number 23

The First Twain Enabled Image Scanning & Viewing Utilities for OS/2

ATTACH

FEATURES:

Fast Image viewing • Easy Screen Capturing • Quick File Conversion • Efficient Color Modification • Powerful Image Manipulation • Real-Time Image Navigation

CURRENTLY SUPPORTS:

Logitech Scanman 256 • Hewlett Packard IIP • Hewlett packed IIC Series



After The *ATTACH* Comes The *REVIEW*

Document Management System

Flexible & Fast Search /Retrieval • Multiple search Fields • Network Support and more...



Solution Technology, Inc. 1101 South Rogers Circle, Bldg 14 Boca Raton, FL 33487
Tel: (407) 241-3210 Fax: (407) 997-6518

Attache and Review are registered trademarks of Solution Technology Inc. All other brand names are trademarks of their respective owners. ©1994 Solution Technology Inc.

Circle Reader Service Number 25

TCP/2™

TCP/IP for OS/2®

Supports *all* released versions of OS/2
including 32bit API for OS/2 2.0 and above

**Our customers
say our tcp/ip is the best.
It's faster, more reliable,
and more complete.
Find out for yourself!**

A DAN LANCIANI PRODUCT
For further information contact: Essex Systems, Inc.
One Central Street • Middleton, MA 01949

(508) 750-6200

TCP/2 is a trademark of DLD consulting. Ethernet is a trademark of Xerox Corporation. OS/2 is a registered trademark of International Business Machines Corporation. VT102 is registered trademark of Digital Equipment Corporation. TCP/2 is based in part on work done by the University of California Berkeley.

Available through OS/2 EXPRESS

Circle Reader Service Number 24

The easy to use 32 bit OS/2 PM IPF Language Editor...

IPF Editor

IPF Editor: \$150
Voice Support Module: \$50

- Add help to applications in minutes
- Designed for easy use by programmers and non-programmers
- Generate C and Resource source files to add help to applications automatically
- Import wordprocessing files, including WordPerfect, quickly and accurately
- Preview IPF output without compiling
- Create all hypertext links automatically
- Optional Voice Support Module
- Includes complete on-line help and documentation

Orders:
1-800-IPF-7622

Information: (206)428-5025
on compuserve at
70410,2416

Perez Computing Services
4725 Monte Vista Place
Mt Vernon, WA 98273

Circle Reader Service Number 26

Seek Professional Help

Strategic Solutions to Software Development

Our experienced staff creates customized software quickly & inexpensively on all major PC operating systems, and specializes in migrating software. Call us at 512-388-7880 for a FREE consultation.

- Software Migration
- Client/Server
- Image Processing
- Custom Software
- Consulting
- Comm. Software

FSI

Final Solutions, Inc.
9114 Brimstone Ln.
Austin, TX 78717

- Multi-Platform
- OS/2
- Windows
- Windows NT
- SOM
- OLE 2

Phone: 512-388-7880 • Fax: 512-388-6015 • Email: finsol@ix.netcom.com

Circle Reader Service Number 27

TO ADVERTISE IN THIS SECTION CALL: KRISTIN MORGAN @212.626.2498

FAX for OS/2

From the developers of **FaxWorks OS/2** and **PMfax**
Client/Server API and Printer Driver Toolkits
Support for LANs, up to 96 lines per CPU,
and all popular fax hardware

Keller Group Inc.

Voice: (612) 429-7273

CIS: 72120,3404

Faxworks is a trademark of SoftNet, Inc.

Fax: (800) 329-3293

Fax: (612) 653-1987

PMfax is a trademark of Keller Group

Circle Reader Service Number 28

SMALLTALK Tools

WindowBuilder™ Pro is an interactive tool that lets you build polished user interfaces quickly and easily in Digitaltalk and IBM Smalltalk, in Windows and OS/2. WidgetKits™ provide additional high-level controls, such as spreadsheets, business graphics, table panes, and CUA'91 controls. No royalties for end-user applications, 90 days free support, and 30-day money-back guarantee. Call for FREE info.

“WindowBuilder Pro/V... the difference between success and failure in Smalltalk/V”

Milan Sremac, President, Medical Software Systems

Objectshare Systems, Inc. v: 408-970-7280 f: 408-970-7282
5 Town & Country Village, Suite 735, San Jose, CA 95128

Circle Reader Service Number 31



REMOTE LAN. NETWORKING

RemoteVision extends LANs transparently over dial-up connections. RemoteVision provides access to a LAN from remote machines or remote LANs. Remote DOS, Windows, and OS/2 machines have a simulated network adapter.

Communications to the LAN is provided by the RemoteVision Server and async modems. Most LAN protocols and applications work, including NetBIOS, Named Pipes, APPC, 802.2, TCP/IP, and IPX. CALL (415) 965-8607 and use option 2 for a free fax-back.

Circle Reader Service Number 29

MISSING SOMETHING?

**Complete back issue library available today!
Every back issue of OS/2 DEVELOPER can be
yours.**

**Call today and complete your library:
1-800-WANT-OS2**

The Next Wave of Lotus Notes Applications Is Here...

Jagre, Inc. provides you with a whole new way to increase your daily work productivity using **IBM OS/2**-based desktop applications and **Lotus Notes**!

Welcome to **Jagre SmartLink**, the easiest quickest way to enable your **OS/2**-based desktop applications to work with **Notes**!

Jagre SmartLink is a set of nine Notes-enabled tools grouped together to provide you with increased productivity for **Document Management, E-mail, Fax, Paging, OCR, Audio, Video and User-based Agents**.

For more information about **SmartLink**, call **Jagre** at (617) 424-8302 or write to 102 Gainsborough St. #202E Boston, MA 02115

Jagre

Circle Reader Service Number 30

SUBSCRIBE TODAY!

Only \$39.95 for a full year subscription to the premier source of tools and techniques for the OS/2 software developer: **OS/2 DEVELOPER**

Call today and don't miss a single issue:

1-800-WANT-OS2

ASIAN OS/2 DEVELOPER SUPPORT



Get help porting OS/2 applications for Japanese, Chinese, or Korean.

Experienced guidance from the leading double byte character set OS/2 developer. Training or consulting on either an hourly or long term basis.



MicroBurst, Inc.

9035 Shady Grove Court
Gaithersburg, MD 20877

(301) 330-2995

Fax: (301) 330-8609

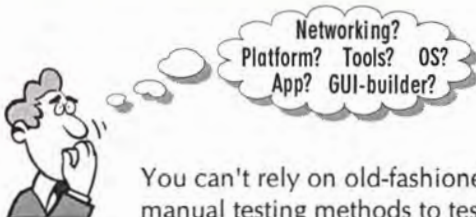
CompuServe: 70334,3616

Internet: 70334.3616 @

COMPUSERVE.COM.

Circle Reader Service Number 34

If you're developing client/server systems, there's one more question you should be asking: How am I going to test them?



You can't rely on old-fashioned manual testing methods to test complex, client/server applications. You need to automate your testing with a tool designed for client/server. ATF is the tool-of-choice for over 100 corporations building client/server applications under OS/2, Windows, and NT. To learn more about ATF, call 617-576-2257. (Or FAX 617-864-7747.)



The Softbridge
Automated Test Facility

Softbridge, Inc. ▲ 125 CambridgePark Drive ▲ Cambridge MA 02140

Circle Reader Service Number 35

ADVERTISER INDEX

Reader Service	Company Name	Page
19	Artisoft	37
15	Athena Design	31
41	Borland International	COV4
5	Compuware Corporation	7
39	Creative Systems Programming	77
17	D-Soft Development, Inc.	32
14	Development Technologies, Inc.	27
24	Essex Systems, Inc.	62
27	Final Solutions, Inc.	63
38	Gpf Systems, Inc.	69
6	Great Lakes Software	9
	Hockware	65
	IBM	13
	IBM	80
	IBM	COV3
30	Jagre, Inc.	63
28	Keller Group	63
22	M. Bryce & Associates	49
2	MetaWare, Inc.	1
18	MHR Software & Consulting	33
34	Micro Burst, Inc.	64
3	Micro Focus	2
8	Mitnor Software	14
31	Objectshare Systems	64
9	On-Line Data	15
1	One Up Corp.	COV2
23	Opt-Tech Data Processing.	63
26	Perez Computing Services	63
7	Pinnacle	11
12	Poet Software	19
4	Programmer's Paradise	4
37	Quadron	68
11	SE International	7
35	Softbridge, Inc.	64
10	SofTouch Systems, Inc.	16
16	Solution Technology, Inc.	32
25	Solution Technology, Inc.	63
36	Solution Technology, Inc.	68
21	Synaptec	48
29	Token Technology, Inc.	64
20	Watcom	39

OS/2 Programming Shouldn't Have to Hurt.



Introducing VisPro/C and VisPro/C++,
easy-to-use tools for IBM C Set and C Set ++
that aren't hard on your wallet.

Other tools allow you to build robust, 32-bit OS/2 applications, but they also require more of your time and money. VisPro/C and VisPro/C++™ give you complete drag and drop visual programming, with all of the bells and whistles -- without having to reach deep into your pockets.

Bells and Whistles

VisPro/C and VisPro/C++ pack an unbelievable amount of functionality into a box. You get a Workplace Shell-enabled GUI environment that fully supports the IBM C Set compilers and User Interface Class Library. Both products have the most CUA '91 objects from simple buttons to 3-D business graphics and everything in between.

And if that isn't enough, we've included a visual DB2/2 database designer that allows you to create embedded SQL client/server applications or reverse-engineer existing DB2/2 databases.

From One World-Class Tool Comes Two More

VisPro/C and VisPro/C++ follow in the footsteps of VisPro/REXX, the pioneer visual REXX programming tool. Thousands of customers already rely on VisPro/REXX for powerful OS/2 development.

Now we provide the same renowned functionality for the C and C++ languages. At the same easy-on-your-wallet prices.

See it to believe it!

Buy now and take advantage of our no risk, 60-day money back guarantee.

Tool	Price
Gpf 2.0 Single Platform	\$495
Gpf 2.1	\$1,295
Kase: C++	\$1,495
Kase: VIP	\$3,495
VisPro/C or C++	\$399

**\$199 until
12/15/94**



HockWare Incorporated
P.O. Box 336
Cary, NC 27512-0336
919-380-0616
919-380-0757 FAX
Go HockWare on CompuServe
hockware@vnet.net on Internet

HockWare, VisPro/C, VisPro/C++ and VisPro/REXX are trademarks of Hockware Incorporated. All other company, product and brand names are trademarks and/or registered trademarks of their respective holders and are mentioned for reference purposes only. ©1994 HockWare Incorporated. All rights reserved.



Many of you have asked for an update on the progress being made at Taligent—the company's impact on desktop software in general and on the development and migration of OS/2 applications in particular. This is the first of what we hope will be a series of articles about the Taligent Application Frameworks.—Ed. By DEAN RODDEY

A Peek at Taligent: The Graphics Subsystem



Dean Roddey

This article provides a broad overview of the graphics subsystem of the Taligent Application Frameworks. As everyone has probably heard by now, IBM and Apple have been working together to create the next generation of operating systems. The two companies founded a third company named Taligent, which is charged with combining the strengths of both parent companies to create a fully object-oriented operating system. Unlike current operating systems, which provide a layer of classes over a procedural kernel at best, Taligent will be object oriented all the way from its foundations to the user interface.

Such a system represents a massive shift in operating system technology, of course. So given the realities of marketplace inertia, IBM has decided to release the upper (application development) layers of the Taligent system as an application development environment that runs over OS/2, AIX, Workplace OS/2, and so on. This strategy will allow developers to gain the benefits of the power and portability of the Taligent system. And since (from the application perspective) these classes are Taligent, applications written to the Taligent Frameworks today will port forward to Workplace OS/2 and on to the real

Taligent system (perhaps in 1996) with minimal modifications.

The Taligent Frameworks are in the prebeta test phase (at press time). The frameworks require the IBM CSet/2++ development system and, at this time, have somewhat high disk requirements. The OS/2 Developer's toolkit is not necessary since the frameworks themselves encompass that functionality.

THE GRAPHICS SUBSYSTEM

Like OS/2 (or any other graphical operating system), the Taligent system has classes for run-time/kernel, windowing, interprocess communications, and graphical output/device control services. This article covers only the graphics subsystem, a voluminous topic itself whose surface can barely be scratched in the space available.

The graphics system covers primitive graphical output mechanisms such as lines, boxes, and curves; device control for controlling graphical output devices such as the screen, printers, and plotters; attribute control such as colors, line styles, and fill patterns; font control; and text manipulation. And, in a full-featured system like Taligent, it would cover higher-level graphical editing services that would fall more



into the application level under current PC operating systems.

A BRAVE NEW WORLD

The first bit of culture shock occurs when you see the Taligent API; it is really object oriented. In other words, it is not just some light classes among mostly procedural code. Everything is a class, down to such things as lines, curves, and attributes. This should not come as a surprise given the previous description of the eventual Taligent operating system, but it is probably more object oriented than most current C++ systems. This means, of course, that applications under Taligent are created by deriving functionality from existing Taligent classes, not by writing to a procedural operating system or "C" runtime API.

Developers who have worked with large class libraries may have gotten a taste of this, but they were always free to run off and write raw code as needed. But when the operating system itself is object oriented, there is nowhere to run. Taligent freely uses advanced C++ mechanisms such as multiple inheritance, templates, and exceptions. Those classes described later in the article that begin with M are classes that are mixed into other classes via multiple inheritance. Taligent does not support runtime-type identification at this time, probably because RTTI has not yet been officially blessed by ANSI. Also, the Taligent classes are not all derived from a single primeval class. It seems there are around 10 to 15 individual root classes, though this factoid does not come from some definitive statement in the documentation, just from my poking around.

Smalltalk developers may feel free to laugh condescendingly at this point, but for rest of us, a new day is dawning. All of us "roll your own" types who

came up from the wild, wild west of DOS (and even under OS/2 to some extent) might decry the loss of personal creativity, but it is for the best in the end. It also means that the best (application level) developer will not necessarily be the one who has memorized the OS/2 technical reference library. Instead, he or she will be the one who best understands the available Taligent classes and where to derive from them to get the desired functionality for the least effort.

As a small aside, Taligent code seems to avoid the often-abused convention of abbreviating type names. All type names seem to be fully spelled out and can get quite long, with some extreme examples over 40 characters. Some might consider this to be the other extreme; but if you have to err, I guess it is better to err on the wordy side since it is easier to type a long name than to try to develop a psychological profile of the developer's abbreviation tendencies. And if you are going to spell them, you have to spell them all to be consistent.

THE COORDINATE SYSTEM

One small but important change for OS/2 Presentation Manager programmers is that the Taligent system uses an "upper left" corner origin for 2-D graphics. Unlike Presentation Manager, which has 0,0 at the lower left corner and positive offsets to the upper right, Taligent uses the upper left corner, and positive offsets go to the lower right. This is more in line with the rest of the world, but Presentation Manager developers may have to lie on their sides for a while until they get used to thinking that way. Another side effect of this system is that rotations (of fonts for instance) go clockwise, not counter-clockwise as in Presentation Manager. This system is better for textual output since we read downward, but it is not as natural for something like a bar graph in which



greater values are almost always higher.

Taligent also provides a powerful 3-D graphics system, which means that there will finally be a powerful 3-D graphics API for the rest of us—one that is built into a standard, portable operating system. The 3-D coordinate system is the usual right-handed system, in which the positive z axis points out of the screen and the positive y axis points upward. If you are not into 3-D graphics, the “handedness” of a system indicates which direction a positive rotation around an axis will be. If you point your thumb along the axis in the positive direction, the positive direction of rotation is the direction in which your fingers

point. So in a right-handed system, positive rotations are in the counter-clockwise direction. In most 3-D systems, objects are defined in a right-handed coordinate system, but the viewing system flips the system around so that larger z coordinates go into the screen, providing a natural model of the real world in which larger distances are further away.

One sign that the Taligent system is serious is that graphical coordinates are double-precision floating point values. Even developers of Presentation Manager applications might not realize how desperately OS/2 attempts to avoid floating point math. Floating point numbers must be manipulated as separate integer

and fractional portions in a fixed structure. There is not even a toolkit define for a floating point type, except in the System Object Model headers. This is for the (supposed) ability for OS/2 to run on systems without a floating point processor. As this is less and less likely to happen these days, I think it is high time we moved to a floating point representation. Units in this system are in arbitrary “World Coordinates” of $1/72$ of an inch increments. A box that is 72.0 by 72.0 units will be displayed at 1 inch on any system.

THE BASIC GRAPHICS CLASSES

The basic 2-D graphical output class is MGraphic. MGraphic is a specialized container object that

BARCODE ANYWHERE™ for OS/2

Are you seeking ways to increase your productivity and data accuracy while lowering your overall cost of data collection? Look no further than BARCODE ANYWHERE. for Automatic Document Indexing Solutions.

Features:

- Any Angle
- Any Background
- Over 60 Pages per Minute
- Internal Consistency Checks
- Developers Kit also Available
- Pop-up Editor for Invalid Barcodes
- Optional Zoning for Faster Throughput

The only barcode we can not read is one which is pasted on the back side of a page



Barcode Types

- Code 39
- Code 128
- Patchcode
- Code 2 of 5 4Q94
- EAN Codes 4Q94
- UPC Code 4Q94

Input Sources

- Scanner
- Fax
- Focal Plane Array
- Video Cameras

Solution Technology, Inc.

1101 S. Rogers Circle, Building 14, Boca Raton, FL 33487
phone: (407) 241-3210 fax: (407) 997-6518

BARCODE ANYWHERE is a trademark of Solution Technology
OS/2 is a trademark of IBM corporation

Multiple protocols. One slot.



Use Quadron software to save PC expansion slot space and money by running two or more protocols on a single co-processor card. Each port can have a different protocol and line speed.

Async • Bisync • HDLC/SDLC X.25 • LAP-B • Custom protocols

We currently support OS/2, DOS, UNIX and AIX. With little or no modification, you can port your applications from one to another. Call today to find out how we can help you communicate better.



Quadron

209 East Victoria Street, Santa Barbara, CA 93101

805-966-6424

Circle Reader Service Number 36

Circle Reader Service Number 37

holds graphics primitives such as lines, ellipses, and text. The **MGraphic** class provides methods to draw, rotate, scale, and translate all of its contained graphics primitives. Also, methods are provided to control attributes and bounds checking. An **MGraphic** may be rendered by passing an output device object to its **Draw()** method or by passing the **MGraphic** object to an output device object. The attributes used to draw the **MGraphic** depend upon the method used to render it. When the **Draw()** method is called, the **MGraphic**'s attributes are used; otherwise, the attributes of the output device are used.

The beauty of such a system is that the basic graphic object is well defined and, through the magic of

C++, its virtual method API may be extended to cover many different variations without any need to modify existing code. So the developer can derive his or her own classes from **MGraphic** that will become integral graphics objects just as any system-provided objects. For instance, you might derive a class called **MyLogo** that draws your company logo. Objects of this class can then be manipulated in a totally generic fashion by operating system code or third-party code that has no idea what a logo is and doesn't care. The logo can be scaled, translated, rotated, and drawn by this foreign code without any knowledge of the logo's contents.

The **TGraphicGroup** class handles

hierarchies of graphics primitives. A graphic group object can contain **MGraphic** objects and other **TGraphicGroup** objects. Like **MGraphic** objects, **TGraphicGroup** objects provide methods to rotate, scale, translate, and draw their contents. These methods just recursively apply the same operation to each of the group's members. The members of a group may be iterated via a **TGraphicIterator** object. Such a system very cleanly handles many common scenarios in graphics applications. For instance, a home design application must handle nested groups of objects such as the architectural elements that make up a room. If the room is rotated, the objects in the room must be rotated in the same way to stay



Roll A Sure Winner With Gpf!

Visual Programming Tools Are Not Created Equal!

The GUI experts at Gpf Systems offer the best PM GUI development tools available at any price.

Independent experts agree:

"If you are doing serious PM programming and you are not using Gpf, you are working too hard"

Rex Conn, PC Magazine

"Gpf not only saved coding and debugging time, but actually made writing PM applications easy and enjoyable. If you need to get your Presentation Manager program up and running in the shortest amount of time, Gpf is the tool to use"

Steve Mastriani, OS/2 Professional

Gpf 2.1 Professional

- 3 GL source code generation - C, C++, (PL/1 optional)
- Cross platform - OS/2 2.x, OS/2 1.3, MS/Windows

GpfRexx

- EXE generation - from WYSIWYG point & click design
- REXX language for custom logic

Gpf Development Tools

- Are compatible - generate 3GL source from REXX prototypes
- Fully support CUA 91 - Drag/Drop, NoteBooks, Containers, etc.
- Support MPM - Multi-Media and Multi-Media controls
- Design Client/Server - Built in SQL, DB2-OS/2 and DDCS
- Impose **NO ROYALTIES** - ever

Stop Gambling, Try Us*

*For FREE Demo Software Call or Fax



Gpf
SYSTEMS, INC.

Phone: (203) 873-3300

Fax: (203) 873-3302

Sales: (800) 831-0017

30 Falls Road, Moodus, CT 06469





in the same relative positions. Instead of manually moving through a list of elements and rotating each one, a Taligent-oriented system would just rotate the room. This operation would kick off a recursive traversal of the entire hierarchy of objects, applying the same rotation to every element of the scene. Of course, your own derivatives might create specialized effects by modifying the reaction to the methods, such as multiplying an applied transformation by some factor.

At a somewhat higher level is the `TPicture` class, which provides capabilities somewhat akin to a Presentation Manager metafile, that is, a list of graphic primitives that are manipulable and render-

able as a cohesive unit. Unlike a `TGraphicGroup`, `TPicture` does not contain `MGraphic` objects. Instead, `MGraphic` objects are rendered to the `TPicture`, which copies them into an efficient internal format. The original `MGraphic` objects are not adopted or changed and are still the responsibility of the calling code. `TPicture` objects cannot be rendered to a `TGrafPort` object. Instead they are rendered to the higher-level `TPicturePort` objects, which understand the `TPicture` format. Also, like metafiles, the individual primitives rendered to the `TPicture` are not individually accessible anymore. The `TPicture` must be manipulated as a whole, thus affecting all of the related primitives.

ATTRIBUTE CLASSES

Drawing attributes are handled via the `TGrafBundle` class. Each primitive object or output device object may "adopt" attribute bundles to control drawing attributes. Attributes in a hierarchy of graphic objects are handled very nicely. If a graphics object does not explicitly provide a particular attribute, the parent group object will provide the default attribute. Color support is extensive in Taligent and is discussed separately later.

FONT CLASSES

In Taligent, a character is called a glyph. A string of characters for output is called a glyph run and is represented by the `TGlyphRun` class. As a derivative of the basic graphics classes, it can be scaled, rotated, and translated just like any other primitive. It also adds methods to control the origin of the glyph run, query the origin of each glyph via the subscript operator, and control the font used to display the run. More advanced font operations such as kerning control are provided at a higher level to avoid burdening all applications with the associated overhead.

A `TGlyphRun` has an associated `TFont` object that defines the font used to render it. `TFont` is like a logical font id in the Presentation Manager Graphical Programming Interface (GPI). `TFont` offers a static method for enumerating the available fonts that meet a particular set of criteria. Virtual methods are provided that give the information needed to correctly space successive glyphs or glyph runs. This is a powerful concept since it means that the developer can derive a font class that provides specialized spacing and pass it to code that knows nothing of the application's specialized needs.

As far as I can tell, the Taligent system does not make use of the native hardware fonts on the hard-

```
void MAIN()
{
    TGrafDevice *fDevice;
    TGrafPort    *fImagePort;
    WINDOW      fWindowId;
    // Create a window, output device, output port
    TGPoint      pntLeft(0, 0);
    TGPoint      pntWndSize(800, 600);
    fWindowId    = CreateWindow(pntWndSize);
    rectSize     = TRect(pntLeft, pntWndSize);
    fDevice      = new TZGrafDevice(rectSize, fWindowId);
    fImagePort   = new TRootGrafPort(fDevice);
    // Set up the color
    TGrafBundle *bundle
= new TGrafBundle(new TColorPaint(TRGBColor(1,0,0)) , new
TColorPaint(TRGBColor(0,0,0));
    TPolygon     polyDemo(TGPolygon(5, TRUE), bundle);
    // Set the polygon points
    polyDemo.SetPoint(0, TGPoint( 0.0, 0.0));
    polyDemo.SetPoint(1, TGPoint(100.0,0.0));
    polyDemo.SetPoint(2, TGPoint(100.0,100.0));
    polyDemo.SetPoint(3, TGPoint( 0.0, 100.0));
    polyDemo.SetPoint(4, TGPoint( 0.0, 0.0));
    // And draw it on the port polyDemo.Draw(*fImagePort);
    // Clean up the port and device
    delete fImagePort;
    delete fDevice;
}
```

Figure 1. A program, minus a lot of the practical infrastructure, that will draw a polygon.

ware to which it will be ported. The obvious reason is that the native OS/2 or NT font systems cannot provide Taligent with the information it would need to transform the text in three dimensions. Serious graphics packages such as Corel Draw have private font systems for the same reason.

COLOR CLASSES

Taligent supports many different color models, including CYMK, RGB, Gray Scale, HSV, HSL, Spectral, XYZ, and YIQ. Each device provides a color profile to the system that defines its ability to produce colors in these models and allows the system to provide color matching between models. The lingua franca of colors are the RGB and XYZ models. All other color model classes must be able to convert to and from RGB or XYZ colors.

Taligent color classes also support an opacity value, which controls how opaque the color is. Its default value in the color class constructors is to make the color fully opaque since that would be the desired value the majority of the time.

DISPLAY DEVICES

Under Taligent, a graphical output device is represented by a `TGrafDevice`. Hardware manufacturers would provide a derivative of this class for their hardware device, so `TGrafDevice` is the object-oriented version of a device driver. Rumor has it that some variation of this system will make it into the Workplace OS micro kernel but not in the original incarnation.

Graphical output is accomplished by outputting `MGraphic`-derived objects to `TGrafPort` objects. In general, there would be no need to derive classes from `TGrafPort`; however, it might be necessary or advantageous for some applications to do so to control the default attributes or actions of a device. So

basically, a `TGrafPort` object is very roughly analogous to a presentation space under the Presentation Manager GPI.

ADOPTING ORPHANS

One of the major sources of memory leaks in any system is caused when a called function/method dynamically allocates a new object and returns it to the caller, who then becomes responsible for deleting the new object. If the caller shirks this duty, a (hopefully serious) memory leak will occur. If it is not serious, then it might never be caught during testing and will just lead to mysterious workstation problems in the field as the application eats up memory and the swapper eventually fills the disk. The other extreme is just as bad, where a new object is passed to another object that expects to gain exclusive control over this new object and to handle its eventual destruction. If the caller does not realize this and continues to use the object, it may be destroyed (or modified) behind the caller's back.

A leading contributor to this leakage is that there is nothing in the language semantics that indicates unambiguously that such an event is occurring. It is up to the developer to document the function and the user to read the documentation (ha!). The Taligent system uses a consistent naming model that addresses this issue. A method that returns an object for which the caller is responsible will have the word `Orphan` in the name. For instance, `OrphanStr()` would create a new string object and return it. A method that accepts an object and expects to then control it will have `Adopt` in its name. So `AdoptStr()` would accept a new string and become responsible for it.

Of course, this is all just semantics, but it helps. Both situations can continue to occur, but at

least it will be more apparent to the developer where the problem is if the method name encodes its effect. This is a system that I will probably adopt for myself, even for non-Taligent code.

SUMMARY

On the whole, I find the Taligent Frameworks to be quite exciting and very full-featured. In fact, it may be too full-featured for some. Such high-level, well-integrated functionality (and a double-precision floating point coordinate system) does not come for free. I would think, though, that the high-end, large-scale applications that would be obvious early targets for Taligent would already have stringent enough hardware requirements to live with the extra overhead. I think that as hardware increases in power over the next year and new, more powerful RISC chips, such as the Power PC, bring us greater performance for lower prices, the market will be more than willing to sacrifice some CPU cycles for the portability and operating system independence that the Taligent system provides.

I have always shied away from large class library systems because of the fact that they inevitably leave something out, take a least common denominator approach to portability, do not support some crucial operating system functionality, or lag behind the most recent release of the operating system. But when the class library is the operating system, I can no longer see any contrary argument. The class library developers will have to supply full functionality because that will be the only means of executing applications, and they cannot (by definition) fall behind the operating system releases. Also, there will not be the extra inconvenience, disk requirements, and royalties of third-party libraries. Instead, a rich and stan-





dard environment will be on every desktop waiting to be put to use.

Of course, some will always complain of too steep a learning curve, the pains of developer retraining, and so on. But, the fully object-oriented hill must eventually be climbed anyway. If you can climb it now and spend the next

four to five years in relative comfort on the next plateau, it seems to make sense to do so. For OS/2 developers, the promise of writing now to a class API that will provide OS portability (and eventually become an integral part of the future grail of operating systems) should be a powerful incentive.

Dean Roddey is a senior engineer at Quantitative Medicine in Annapolis, Md. QMI, a subsidiary of Marquette Electronics, creates OS/2-based clinical information systems.

Dean is also the well-known inventor of the "Intravenous Mr. Coffee." He can be reached through CompuServe at 72170,1614.

Smalltalk

A Special Report from the Publishers of OS/2 Developer!

☒ **YES!**

Please send me _____ copies of the Smalltalk!
Each copy is only \$9.95 in the US or \$10.95 in Canada.
Please add \$2.50 for shipping and handling.

☐ My check is enclosed

☐ Charge my credit card: Visa / MasterCard / American Express

Name _____

Signature _____

Date _____

Send them to:

Name _____

Address _____

City _____

State _____

Postal Code _____

Mail:

Smalltalk
P. O. Box 7046
San Francisco
CA 94120

Phone:

1-800-444-4881

Fax:

(415) 905-4967

Order now!

Quantities are limited.

It's A Fact.

The one place to be for software developers this winter is CES®. It's where today's developers, vendors and buyers will converge to capitalize on a new generation of software and hardware for playback, production, imaging and display—in time to make 1995 even more profitable. Entertainment, infotainment, edutainment, education, business, interactivity.

.....
CES.®

What the world's coming to.

J A N U A R Y 6 - 9 , 1 9 9 5

Sounds, images, actions. It's all at CES, the event of the year. Fact is, all you have to do is fax the coupon below to develop a killer strategy for success now and in the future. Send your response today!



**1995 INTERNATIONAL WINTER
CONSUMER ELECTRONICS SHOW®**

**FRIDAY, JANUARY 6 TO
MONDAY, JANUARY 9, 1995**

LAS VEGAS CONVENTION CENTER

LAS VEGAS HILTON ■ THE MIRAGE ■
SAHARA HOTEL

LAS VEGAS, NEVADA USA

FAX NOW TO RECEIVE YOUR BROCHURE—202/466-4188

133

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone _____ Fax _____

☐ I am a Retailer/Buyer/Service Professional

☐ I am a Manufacturer/Rep/Consultant

If you prefer, write to: **CES**, 2001 Pennsylvania Avenue, NW, Washington, DC 20006-1813 USA



Buyer's Guide

The staff of OS/2 Developer put together this special section to guide you through the various graphical user interface and graphical programming interface tools available for OS/2. The products described in this guide are based on surveys sent to vendors; they are provided as a service to you and are free to vendors. These listings do not represent an endorsement by OS/2 Developer. Although every effort has been made to ensure accuracy, we do not assume any responsibility for error or omission in this section.

Graphics Buyer's Guide

In the September/October issue of OS/2 Developer, we incorrectly listed the prices for ParcPlace Systems' VisualWorks 2.0. The price for Windows, NT, Mac, and OS/2 is \$2,995.00, not \$429.95. The price for UNIX is \$4,995.00, not \$49.95.

ARS NOVA

Circle No. 101

NEDT is a tool for designing complex graphical editors. The areas where NEDT can be implemented range from graphical information systems to generic diagram editors and electronic circuit design. A NEDT graphical interface can be piggybacked into an existing application, and functions such as grouping, zooming, and drag-and-drop of graphical objects are accessible. It is available for use on Windows, OS/2, NT, Mac, Sequent, HP 9000, SUN OS, Solaris, and IBM 6000. Price: \$1,800.

BLACK-BOXES is a generic graphical editor. It facilitates the development of editors for diagram structures by providing the user with a ready-to-use editor structure that can be plugged in to applications by specifying a number of interface messages. Potential applications include petri net, chemical process simulators, factory layouts, and electronic circuit design. It is available on Windows, NT, OS/2, Mac, Sequent, HP 9000, Sun OS, Solaris, and IBM 6000. Price: \$3,500.

ARS NOVA Software GmbH, Stettener StraBe 32/3, 73732 Esslingen Germany, 49 711 371 4001, fax 49 711 370 4003.

AUTUMN HILL SOFTWARE INC.

Circle No. 102

BD/CX 3.0 is a baby driver interface library for the C and C++ developer who wants to provide integrated printer support from an application. The kernel supports over 800 printers and allows printing of text and graphics on a supported printer. It also allows the user to control

scale, rotation, position, palette, and dithering. Price: \$499.

Menuet/CX 4.0 is a multiplatform graphical user interface application framework for C++. It provides true single-source capability. Features include full CUA compliancy, formatted data entry fields, and a range of graphical user interface controls. There are prebuilt dialogue, font, and icon editors. The graphics file I/O uses BMP, window PCX, and TIFF formats. Price: \$599.

Autumn Hill Software Inc., 1145 Ithaca Dr., Boulder, Colo. 80303, (303) 494-8865, fax (303) 494-7802.

ART IN APPLES

Circle No. 103

ArtGUIDE 1.0 is an application development environment for design and implementation of complex multi-user information systems in VisualWorks. It supports the analysis and design phase, automated generation of classes, and graphical user interface design and specification. Graphical user interfaces are generated in the run time from specification, reflecting the current state of the object and its structure. Graphical user interfaces allow direct manipulation with objects and their parts, such as drag-and-drop and copy. ArtGUIDE generates reusable frameworks that may be plugged in to complex applications without reprogramming. It is an application development environment and front-end to ArtBASE. Price: \$500.

Art in Apples Ltd., Kremelska 13, 84503 Bratislava, Slovak Republic, 42 7 365 233 362 889, fax 42 7 365 235 777 773.

CIRRUS TECHNOLOGY INC.

Circle No. 104

Image ViewerObject 1.00 provides the capability of viewing and manipulating bitonal Tiff Group 3, bitonal Tiff Group 4, gray scale, and



color JPEG files. The Image Viewer can be provided with a front-end graphical user interface wrapper, which is used to integrate the object into development environments such as Watcom's VX-REXX and IBM's VisualAge. Price: \$1,200.

Scan Object 1.00 is based on IBM's System Object Model (SOM) technology and provides accessibility to a wide range of scanners. It is offered as fully functional APIs or integrated with a development environment. It is adaptable, flexible, and capable of providing complete software control of the scanner hardware. Price: \$2,500.

Cirrus Technology Inc., 5301 Buckeystown Pike, Fourth Floor, Frederick, Md., 21701, (800) 272-1135 or (301) 698-1900, fax (301) 698-1909.

GFT GESELLSCHAFT FÜR

TECHNOLOGIETRANSFER GMBH Circle No. 105

GRITplus 2.2 is a software development tool dedicated to the design of graphical user interfaces. It is available for UNIX, OS/2, and Windows NT. It has interfaces to all common database systems (DB2/2, Sybase, ORACLE, INFORMIX, GUPTA, INGRES) and is portable to major operating systems. The GRITplus product family consists of various tools like CASE, Report-Writer, and Chart and is completely object oriented. Price: \$4,500.

GFT, Leopoldstraße 1, St. Georgen, Germany 78112, 49 7724 9411 0, fax 49 7724 9411 94.

GPF SYSTEMS INC.

Circle No. 106

GpfRexx 1.0 visual OS/2 programming with REXX. Users point and click to create CUA '91 applications and display help. The Gpf Interface Builder handles all of the logic for managing events, windows, controls, and their data. You can take advantage of multimedia, drag-and-drop, multitasking, SQL (DB2/2), and advanced communications (APPC, CPI-C, EHLLAPI). It is royalty-free. Price: \$199.

Gpf 2.1 Programmer's Development Kit includes Gpf 2.1 and GpfTools to provide the complete professional toolkit. There is support for OS/2 2.x, OS/2 1.3.x, and MS Windows 3.0 and 3.1. Users employ point-and-click for CUA '91-compliant graphical user interfaces. It generates C or C++ source, Help source, and all ancillary files. GpfTools is used for manipulation and automatic documentation of Gpf designs. Price: \$1,440.

Gpf 2.0 Single platform is a point-and-click visual programming environment for OS/2 2.x. Complete prototyping, testing, and generation of full CUA '91 designs, including navigation code and context-sensitive help, is possible. There is ANSI C source generation and embedded SQL for DataBase Manager or DB2/2. There are no run time or royalties. Price: \$495.

GPF Systems Inc., 30 Falls Rd., P.O. Box 414, Moodus, Conn. 06469, (203) 873-3300, fax (203) 873-3302.

THE GRAPHICS NETWORK LTD.

Circle No. 107

The Business Graphics Library 2.3 for OS/2 2.x provides a collection of routines for displaying presentation graphics in your applications. This 32-bit application includes various line, bar, and pie charts in 2-D or 3-D, as well as functions for displaying text in boxes with optional shadows, for easy font selection, and for drawing thick lines and polylines. Features may be used on a window or printer. It is supplied as both a DLL and a static library. Price: \$185 or \$775 for the source code.

The Graphics Network Ltd., The Siskins, Hatherop Rd., Fairford, Glos. GL7 4JZ U.K.

GREAT LAKES SOFTWARE INC.

Circle No. 108

GLS-Presto 2.01 is an integrated development environment that allows you to create graphical user interface applications for OS/2 and Windows in COBOL, C, and C++. Presto works with popular compilers including Microfocus Cobol, IBM C-Set++, and Borland C++. No royalties are required. Price: \$395.

Great Lakes Software, 810 East Coliseum Blvd. Ste. 109, Fort Wayne, Ind. 46805, (219) 481-5809, fax (219) 483-8301.

HOCKWARE INC.

Circle No. 109

VisPro/C 1.0 is a full-featured OS/2 2.x visual programming tool for the IBM C-Set and C-Set ++ First Step compilers. It provides drag-and-drop programming for creating royalty-free, multithreaded, 32-bit graphical user interface applications. VisPro/C includes Workplace Shell integration, CUA '91 objects, a graphical class browser, a visual DB2/2 database designer, and a SOM-based object builder. Price: \$399; special introductory price of \$199 until Nov. 30, 1994.

VisPro/C++ 1.0 is an OS/2 visual program-



ming tool for IBM C-Set++ compiler and User Interface Class Library. It provides a drag-and-drop programming environment for creating royalty-free, multi-threaded, 32-bit graphical user interface applications. Workplace Shell integration, CUA '91 objects, a graphical class browser, a visual DB2/2 database designer, and a SOM-based object builder are included. Price: \$399; special introductory price of \$199 until Nov. 30, 1994.

VisPro/REXX Bronze Edition 2.03 has an easy-to-use drag-and-drop environment that allows you to quickly develop 32-bit OS/2 2.x graphical user interface applications. It includes many of the CUA '91 objects, full Workplace Shell integration, multiple development views, and a SOM-based object builder. The Bronze edition provides standalone, royalty-free executable. Upgrade to the Gold edition is available. Price: \$99.

VisPro/REXX Data Entry Object Pack 1.0 is an add-on package of five popular business objects for both VisPro/REXX Gold and Bronze editions. Objects include a formatted entry field, spreadsheet, split bar, calendar, and clock. It is a REXX programming tool that provides extensibility. Price: \$119.

VisPro/REXX Gold Edition is a professional drag-and-drop visual programming tool for 32-bit OS/2 2.x graphical user interface development. It includes all CUA '91 objects plus 3-D business graphics and multimedia, full Workplace Shell integration, multithread support, a visual DB2/2 database designer, multiple development views, a SOM-based object builder, and a standalone, royalty-free executable. Price: \$299.

Hockware Inc., 315 N. Academy St. Ste. 100, Cary, N.C. 27513, (919) 380-0616, fax (919) 380-0757.

IBM CORP. Circle No. 110

Animated Design II is an OS/2 program that animates process models created using the Ward and Mellor design method and the Excelerator II 1.1. AnDes lets you examine the dynamic behavior of the processes you model; it makes their behavior visible by using animated graphics that show the activation and deactivation of processes, the flow of signals and dates, and transitions from one state to another.

IBM Corp., Hanns-Klemm-Str. 45, D-71034 Boeblingen, Germany, 49 7031 166118, fax 49 7031 166901.

Distributed Application Environment (DAE) 1.3 provides tools for developing and running distributed client/server solutions in a homogeneous or heterogeneous platform, network, database, and

graphical user interface environment. DAE includes downstream device support for use with barcode scanners, programmable controllers, robot controllers, and DCS systems. Price: \$445 to \$3,465.

Paperless Manufacturing WorkPlace

1.2 is an OS/2 client/server application that allows you to create, maintain, distribute, and display electronic work instructions. Applications are built by integrating windows of visual information that may contain text, images, and graphics. You can customize a Paperless Manufacturing WorkPlace application to the needs of your business. Price: \$1,485 to \$9,999.

Plantworks 2.2 provides a comprehensive, graphically oriented set of tools for building and running supervisory control and data acquisition and other applications used by manufacturing companies. Key functions include application development using graphical flowchart diagrams, access to an expandable variety of manufacturing devices, animated graphical displays that can be associated with device data, and built-in functions. Price: \$3,895 to \$8,000.

IBM Corp., 1000 NW 51st St., Boca Raton, Fla. 33432, (800) 323-4968 or (407) 443-6595, fax (407) 443-6824.

IMAGESOFT

Circle No. 111
Classworks 1.2 is a C++ class library that abstracts the complete operating system as opposed to just the graphical user interface component. Classworks includes advanced features such as support for systems services, real-time field editing, multithreading, and interprocess communications. Applications developed with ClassWorks are completely portable between Windows 3.1, Windows NT, OS/2, and Unix. Price: \$499.

Object/Designer is an extensible framework for code generation. Using the high-level scripting language, developers can easily modify the source drivers to generate any type of code. Object/Designer allows developers to create custom objects and provides an open API developers can use to manipulate objects from a DLL. Price: \$499.

ImageSoft, 2 Haven Ave., Port Washington, N.Y. 11050, (800) 245-8840 or (516) 767-2233, fax (718) 767-4307.

INSOFT OY

Circle No. 112
Prosacdm Concurrent Document Manager 4.0 implements an automatic support for software documentation in Prosa Case environments. Prosacdm is a hypertext interface between model data bases and desktop publishing systems. Documents can contain both graphic and data dic-

tionary information. The changes made in a model database are automatically reflected in documentation. It is compatible with Ami Pro, Decwrite, Framemaker, Interleaf, Microsoft Word, PageMaker, Ventura, and WordPerfect DTP systems.

Insoft Oy, Prosa Software, Kirkkokatu 5 B FIN-90100 Oulu, Finland, 358-81-376128, fax 358-81-371754.

INTELLIGENT ENVIRONMENTS

Circle No. 113
AM/Impact 2.11 is a graphical presentation toolkit that enables the user to create, display, manipulate, and print a wide range of forms, images, and pictures. Application Manager can print complex forms with graphics and logos and interact with high-resolution pictures of products, geographic sectors, and illustrations. It dynamically creates multipage word-wrapped text documents. Price: \$995 per user.

Intelligent Environments, 2 Highwood Dr., Tewksbury, Mass. 01876, (800) 669-2797 or (508) 640-1080, fax (508) 640-1090.

INTERSOLV

Circle No. 114
PVCS 5.2 is a solution for software configuration management problems. It is comprised of five integrated products (Version Manager, Configuration Builder, Reporter, Production Gateway, and Developers Toolkit) designed for today's complex development environments. PVCS manages all development objects including binary files, text, graphics, and source code. Price: \$599.

Intersolv, 1700 NW 167th Pl., Beaverton, Ore.

MITNOR SOFTWARE

Circle No. 115
PrintScrn 2.0.1 has four utilities that provide capabilities beyond the scope of OS/2 built-in functions. There are nine different methods for selecting the image area. Users can initiate the Print/Capture operation from their programs and can capture up to seven different file formats of Clipboard. Clipboard viewer provides importing, exporting, viewing, and printing of clipboard images and text and can be used as a file browser. There is a Screen Blanker (with nine different display animations) and a Date & Time Display (for time-stamping images). Price: \$115.

Mitnor Software, 28411 E. 55th St., Broken Arrow, Okla. 74014-1765, (918) 357-1628, fax (918) 357-2869.

PARCPLACE SYSTEMS INC.

Circle No. 116
VisualWorks 2.0 is a powerful client/server tool for building portable applica-

tions using object-oriented technology. A database application creator is included for rapid application development. Applications developed are instantly portable across multiple platforms, are scaleable across the enterprise, and can have their functionality distributed between both clients and servers. Price: \$2,995.

ParcPlace Systems Inc., 999 East Arques Ave., Sunnyvale, Calif. 94086, (800) 759-7272 or (408) 481-9090, fax (408) 481-9095.

SAS INSTITUTE INC.

Circle No. 117

The SAS system is an integrated suite of software products for enterprise-wide information delivery providing organizations with tools to access, manage, analyze, and present their data within an application development environment and menu-driven systems. Capabilities within the system include executive information systems, spreadsheets, graphics, data analysis, report writing, quality improvement, project management, client/server computing, database access, and applications development. SAS is modularly designed so organizations can design a system to meet their specific application needs. It takes advantage of OS/2 and can be integrated to run across multiple hardware platforms—LANs, workstations, minicomputers, supercomputers, and mainframes.

SAS Institute Inc., SAS Campus Dr., Cary, N.C. 27513, (919) 677-8000, fax (919) 677-8123.

STEPSTONE CORP.

Circle No. 118

Objective-C 4.3.2 is a language that provides a rich implementation of object-oriented technology. It enables the user to program in a familiar language while retaining the ability to use the power of object-oriented technology. Price: \$495.

Stepstone Corp., 75 Glen Rd., Sandy Hook, Conn. 06482, (800) 289-6253 or (203) 426-1875, fax (203) 270-0106.

TOM SAWYER SOFTWARE

Circle No. 119

Graph Layout Toolkit 2.0 is a portable graph management system that immediately face-lifts applications with its graph layout algorithms. Three layout libraries are available—circular, hierarchical, and symmetrical. These C++ class libraries include ANSI C APIs and allow flexibility for multiplatform graphical user interface development.

Tom Sawyer Software, 1824 B Fourth St., Berkeley, Calif. 94710, (510) 848-0853, fax (510) 848-0854.

TRINZIC CORP.

Circle No. 120

Aion Development System 6.4 is a client/server development environment intended for "run-the-business" applications that incorporate business policies, judgement, or experience and are too complex to develop or maintain productively with traditional 3GLs or 4GLs. Rule-based and object-oriented technologies enable the AionDS developer to solve tough business problems. Price: \$9,000. (Volume discounts are available.)

Trinzic Corp., 101 University Ave., Palo Alto, Calif. 94301, (800) 234-7724 or (415) 328-9595, fax (415) 321-7728.

ZINC SOFTWARE INC.

Circle No. 121

Zinc Application Framework 4.0 is an object-oriented, multiplatform, internationalized, C++ class library and visual development tool. Zinc supports IBM OS/2, Microsoft Windows, OSF/Motif, DOS, Apple Macintosh, and NEXSTEP—with one set of source code. Zinc 4.0 supports 13 languages (single- and double-byte). Product includes the class library, Zinc Designer, and full source code.

Zinc Software Inc., 405 South 100 East 2nd Floor, Pleasant Grove, Utah 84062, (800) 638-8665 or (801) 785-8900, fax (801) 785-8996.



GET FAST ANSWERS TO YOUR DEVELOPMENT QUESTIONS

You need to do two things:

- Go on-line with CompuServe®.
- Use Golden CommPass to do it.

CompuServe hosts OS/2 developer and user forums. IBM developers have responses to anything that's got you stuck. Other OS/2 programmers and enthusiasts are there, too, comparing notes and giving feedback. It's definitely the place to be.

Time is money when you connect to CompuServe, and Golden CommPass saves you both. It lets you compose your questions off-line, send them in a

flash and disconnect. It gets your replies while you're busy programming.

Golden CommPass keeps your development schedule on course. The fact is, the sooner you start using our program, the sooner they'll be talking about yours.



Golden CommPass® OS/2® NAVIGATION SOFTWARE

Creative Systems Programming Corporation
(609) 234-1500 • Fax: (609) 234-1920
Internet: 71511.151@CompuServe.com



Available
and Ready
for LAN Server



Available
and Ready
for OS/2

Circle Reader Service Number 39



New Products for OS/2



TOOLS and UTILITIES

Accusoft Image Format Library 4.0. According to the company, this product can offer longevity, support of popular Raster Image Formats (TIFF, JPEG, PCX, BMP), and cross-platform support (including OS/2).

Accusoft Corp.
Phone: (508) 898-2770

Circle No. 151

Btrv++ 2.51. This is a class library for Novell Btrieve. It has access to Btrieve DDF data dictionaries, an SQL processor, and a Standard Windows DLL (16- or 32-bit). It supports OS/2 with a common interface for complete portability. It comes with full ANSI standard source code.

Classic Software Inc.
Phone: (313) 677-0732

Circle No. 152

GPSS World 1.0. Minuteman Software announces its new general-purpose modeling environment designed for simulation professionals. GPSS can predict the behavior of complicated systems and can handle large industrial models while exploiting virtual memory, 32-bit computing, preemptive multitasking, symmetric multiprocessing, and distributed simulation. The World family contains three products: GPSS World, the basic modeling environment; Simulation Server, which performs remote simulation services; and Simulation Studio, which does hierarchical modeling and animation. Features include drag-and-drop model building, client/server operation, a high-performance model translator, point-and-shoot debugging, an embedded programming language, built-in probability distributions, multiple data types, and tightly integrated discrete and continuous phases.

Minuteman Software
Phone: (508) 897-5662 ext. 542

Circle No. 153

Central Point Anti-Virus. This virus protection features complete 32-bit implementation, Workplace Shell-enabled application, detection of unknown viruses without prior knowledge of file/sick in a clean slate, ability to distinguish between legitimate file changes and changes caused by a virus, generic virus removal, and virus alert to anti-virus NLM on server.

Central Point Software
Phone: (503) 690-8090

Circle No. 154

DeskMan/2 1.5. This multifunction tool is designed to manage, secure, back up, and restore the OS/2 workplace. It provides the advantages of desktop management plus migration of icons, objects, and desktops between home and office machines. This product is designed to make it easier to use and administer the OS/2 system.

Development Technologies Inc.
Phone: (708) 291-1616

Circle No. 155

eXceed 2.0. This 32-bit, X11R5-compliant X server enables OS/2 users to connect to and display applications from UNIX and VMS computers. It bundles an X Window System Development Toolkit, allowing programmers to develop X Window applications for the OS/2 environment.

Hummingbird Communications.
Phone: (905) 470-1203

Circle No. 156

FindOut! This data-access tool helps users navigate data, issue queries, display information, and create reports. It also allows users to design models and database views based on business rules and processes.

Open Data Corp.
Phone: (617) 860-8300

Circle No. 157

Galaxy. This product helps developers build large-scale, distributed applications and move them across platforms without changing lines of code. Once an application is written, you can move it to UNIX, Windows 3.1, Windows NT, OS/2, Macintosh, and Open VMS platforms. According to the company, Galaxy is used on Wall Street and in the Silicon Valley.

Visix Software Inc.
Phone: (703) 758-2711

Circle No. 158

Gamelon File I/O Library. This product is available for OS/2, is written in C++, and is built around the concept of object-based storage. The architecture allows developers to build flexible file structures using combinations of aggregate and data objects that cannot be created using relational database systems. It fea-



tures aggregate object nesting, logical navigation, and automatic object tracking.

Menai Corp. Circle No. 159
Phone: (800) 426-3566 or (415) 617-5730

LinkRight 1.1. This parallel port and serial port file transfer utility is made specifically for OS/2. The package includes a Presentation Manager version, an OS/2 command line version, and a DOS version. The OS/2 versions are 32-bit multithreaded tasks. The LRCLONER program provides an efficient method of cloning an entire OS/2 partition. This product features compression for faster transfers, uses CRC checking to ensure accurate transfers, and provides full support for long filenames and high-performance file systems.

Rightware Inc. Circle No. 160
Phone: (301) 762-1151

MLINK/ACM. Run on UNIX or OS/2 servers, this product can manage the distribution and collection of data to and from PCs as well as UNIX workstations connected via TCP/IP. XCOM for AnyNet provides managed data transport over TCP/IP between MVS and OS/2 servers. RS/6000 and AS/400 support is planned.

Legent Corp. Circle No. 161
Phone: (703) 708-3118

System Architect 3.0. This runs on OS/2 Presentation Manager 2.1 and supports the Object Modeling Technique (OMT)/Rumbaugh method of object-oriented analysis and design.

Popkin Software & Systems Inc. Circle No. 162
Phone: (415) 354-4404

SOM Class Administrator. This OS/2 SOM WPS run-time class administration tool provides a hierarchical run-time SOM WPS class browser. It also features full class administration utilities, including creation/deletion of class instances, class registration/deregistration, and DLL version control. It is useful for developers and systems administrators.

Synaptec Inc. Circle No. 163
Phone: (404) 942-8699

Syotos Premium. The company announces the availability of its new OS/2 backup and disaster recovery software. It offers OS/2 workstation and network server data protection. The disaster recovery utility protects OS/2 workstations and networks from system failure. If the hard disk becomes corrupt, the user installs the recovery disk to reboot the system. Features include a graphical Presentation Manager user inter-

face, recovery for OS/2, multiple event scheduler, and optional device support for autoloaders.

Sytron Corp. Circle No. 164
Phone: (508) 898-0100

TechBridge Builder for OS/2 1.1. Although this product is implemented in Smalltalk, developers actually use a scripting language that follows a simplified COBOL syntax. TechBridge Builder exploits object-oriented technology and is comparable to Powersoft's PowerBuilder and Gupta's SQL Windows. It is designed for OS/2 and its needs, however, such as painters for the CUA style of graphical user interface—object menus, object interaction via drag-and-drop, containers, and icon view of data objects. There is a free 60-day evaluation offer.

TechBridge Technology Corp. Circle No. 165
Phone: (800) 463-8998

UCSD Pascal. The company announces the release of a 32-bit UCSD Pascal development system. This system is integrated into the IBM Presentation Manager Application Programmer Interface, and Pascal developers can use the Presentation Manager technology with access to the OS/2 graphical user interface for the first time. It includes an OS/2-based IDE, compiler, editor, library manager, cross referencer, and library resource toolkit.

Cabot Software Circle No. 166
Phone: (0272) 586644

Watchit 2.0. This product has a full 32-bit Presentation Manager application and drag-and-drop for ease of use. Data collection features include manual control on the desktop or server, dynamic notification of potential server operational problems via Local pop-up, Log to NET.ERR file, Message ALERT, or Generic ALERT. Data analysis has the ability to export data for additional analysis with a spreadsheet or database.

CSN Inc. Circle No. 167
Phone: (203) 233-2951

X:Change 1.1.1. This cross-platform file manager is a fully graphical, 32-bit application that provides programmers with the technology to manage, transfer, and synchronize MVS and OS/2 files with point-and-click and drag-and-drop. It can be invoked on demand or in batch mode. It provides a REXX interface, which enables programmers to automate routine tasks and to perform unattended file transfer.

SERENA International Circle No. 168
Phone: (415) 696-1800

A satellite image of North America, showing the United States and parts of Canada and Mexico. The image is tilted, showing the curvature of the Earth. Several text callouts are overlaid on the image, each with a line pointing to a specific location. The callouts are in white, bold, sans-serif font. The background is a dark, starry space.

**A WOMAN IN TORONTO
OF THE KIDS TO GRANDMA**

**BECAUSE HE GOT WARPED, A STOCKBROKER IN
SAN FRANCISCO SURVIVES A CRASH INTACT.**

**AN ASTRONOMY STUDENT IN OHIO
WARPS ONTO THE INTERNET. AND
DISCOVERS A WHOLE NEW WORLD.**

**AN UP-AND-COMING ARTIST IN SANTE FE WAS AMAZED
THAT HE COULD GET WARPED FOR UNDER \$90.**

**A MAN IN NEW ORLEANS WARPS HIS WINDOWS
INTO POWER WINDOWS.**



**WARPS PHOTOS
IN STOCKHOLM.**

**A MAN IN NEW YORK WARPS HIS COMPUTER
SO HE CAN PRINT A REPORT, READ HIS E-MAIL AND
FAX AN ORDER FOR A BLT. ALL AT THE SAME TIME.**

**IN RICHMOND, A MAN DISCOVERS WARPING
IS SO EASY, HE DOESN'T HAVE TO ASK HIS
11-YEAR-OLD SON TO HELP INSTALL IT.**

OS/2[®] WARP

**Introducing
OS/2 WARP**

The **new** 32-bit,
multitasking,
multimedia,
Internet-accessed,
crash-protected,
Windows[™]-friendly,
easy-to-install,
totally cool way
to run your computer.

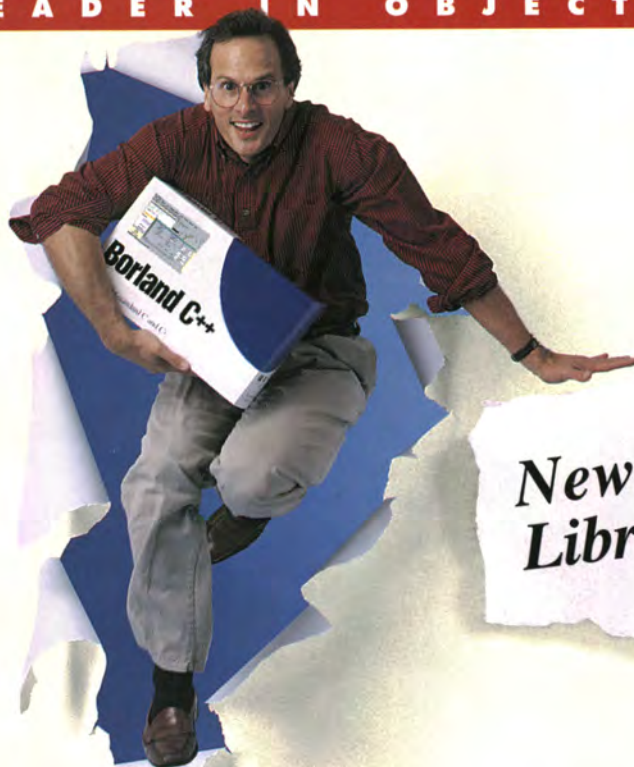
Available for
under \$90.

OS/2 WARP
from IBM.

It's here.

See your local dealer
today. Or, call
1 800 3 IBM-OS2.





**New! ObjectWindows
Library for OS/2**

Get Borland C++ today, and receive OS/2 2.1 free!

There'll never be a better time to try OS/2 development

Borland® C++ is the world's most popular 32-bit compiler. OS/2® is the world's fastest-growing 32-bit operating system. Together in one box, they're the world's best deal.

“The easiest OS/2 environment I've worked in.”

Brett Glass, PC Techniques

32-bit power and Borland C++ flexibility

Borland C++ 2.0 for OS/2 gives you the most flexible Integrated Development Environment (IDE) going. Its advanced visual tool

set, powerful integrated editor, and fully integrated GUI debugger help speed you through even the most complex development project. Together with OS/2 version 2.1 for Windows, you have almost unlimited 32-bit power, and the best tools to develop it.

New ObjectWindows Library for OS/2

Borland's popular ObjectWindows® Library (OWL) is now available for OS/2 Presentation Manager. OS/2 developers will love this fully object-oriented application framework and Windows developers will be able to move their OWL programs to OS/2 quickly and easily.

Limited-time offer

But you'll have to act fast. So see your local Borland languages dealer.

Borland C++ 2.0 for OS/2 with OS/2 2.1. There'll never be a better time to try OS/2 development.

Call now!

1-800-336-6464, ext. 9542

to get everything you need to get started in OS/2 development!

Borland

The Upsizing Company

6362-0001-26

